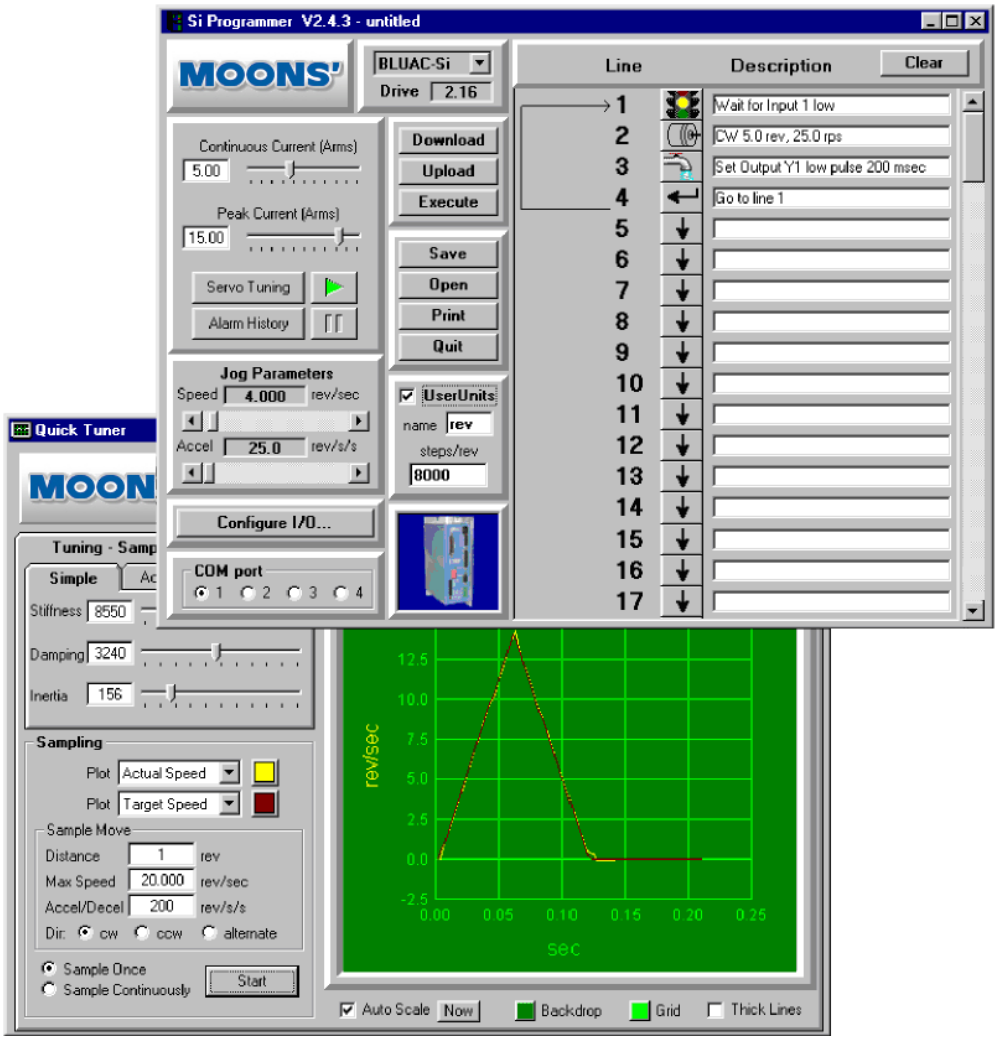
2007-3-14

# Mis Programmer

## Software Manual




for MS1240MI, MS3540MI, MS7080MI

MOONS'

# Contents

**Getting Started** ..... **5**
Installing the Programming Software ..... 6
Connecting to your PC ..... 7
What if my PC has no Serial Port? ..... 7
**Programming** ..... **8**
Which Software Version do I Have? ..... 9
Entering Your Program ..... 9
Copying Instructions ..... 11
Inserting and Deleting Program Steps ..... 12
User Defined Units ..... 12
Front Panel STOP Button ..... 13
Setting the Step Motor Current ..... 13
Idle Current Reduction ..... 14
Setting the Servo Motor Current ..... 14
Microstepping (Step Motors Only) ..... 14
Jogging ..... 15
Configure In puts Dialog ..... 16
Interrupt ..... 16
Quick Decel Rate ..... 18
Limit Switches ..... 18
Motion Output (BLU Servo) ..... 19
Motion Output (STAC6) ..... 19
Brake Output (BLU Servo and STAC6) ..... 19
Special STAC6 Dialogs ..... 20
Servo Faults ..... 22
Front Panel Error Codes ..... 23
Using the Optional MMI ..... 25
Making Your Move ..... 31
MMI Prompt ..... 31
Feed to Length ..... 33
Feed & Set Output ..... 35
Feed & Return ..... 36
Feed to Sensor ..... 37
Feed to Sensor & Return ..... 39
Feed to Position ..... 40
Set Abs Position ..... 41
Save Abs Position ..... 42
Seek Home ..... 43
Wait Time ..... 44
Wait Input ..... 45
Hand Wheel ..... 46
Go To ..... 47
Repeat/End Repeat ..... 48
Reset Loop or Interrupt ..... 50
Set Output ..... 51
If Input Go To ..... 52
Call and Return ..... 54
Change Current ..... 55
Change Servo ..... 56
Comment ..... 57
Command Buttons ..... 58
Download, Upload & Execute ..... 58
Save, Open, Print & Quit ..... 60
Encoder Feedback: 3540i with Encoder Option Board ..... 61
Encoder Feedback: STAC6 ..... 63
**Servo Tuning** ..... **65**

The Motor - Encoder Tab ..... 65
Encoder and Hall Timing - BLU Drives ..... 65
Encoder and Hall Timing - BL Drives ..... 66
Manual Timing Configuration (BL Series only) ..... 67
Encoder Direction (BL Series only) ..... 68
Positioning Error Fault ..... 68
Regeneration Clamp ..... 68
Reset Factory Defaults ..... 69
Alarm History ..... 69
Configure I/O ..... 69
The Tuning - Sampling Tab ..... 69
The Scope ..... 69
Control Loop Tuning - BL Drive ..... 71
Control Loop Tuning - BLU Drive ..... 73
Stiffness Gain Terms ..... 73
Damping Gain Terms ..... 74
Inertia Gain Term: Acceleration Feed forward ..... 75
Filters ..... 75
Monitor ..... 75
Host Terminal ..... 77
BL Tuning Tutorial ..... 77
BLu Servo Tuning Tutorial ..... 81
Getting Ready for Tuning ..... 81
1) Entering a Sample Move ..... 82
2) Start with the KP & KD parameters ..... 82
3) Let's Plot a Move ..... 82
5) The return of KVf and KVff parameters ..... 83
6) Adding in the KAff parameter ..... 84
7) Finishing off with the KI parameter ..... 85
8) Verify the Drive Current ..... 86

# Getting Started

Thank you for purchasing an Applied Motion Products Mis product. We hope you will find that the performance, price and ease of programming make our products the best value for your application.

The Mis Programmer software is used in many Applied Motion products, including the 1240i, 3540i, 7080i, Mis2035, Mis3540, Mis5580 and STAC6-Mis step motor drives, the BLMis7080, BL7080i, BLU100-Mis, BLU200-Mis and BLUAC5-Mis digital servo drives and the Mis-100 indexer. This manual explains how to install the Mis Programmer software and how to program your Mis product.

For information regarding your specific Mis hardware, such as wiring and mounting, please read the hardware manual for that product. Hardware manuals are also included on the CD, at our web site or by clicking the Help button in the software.

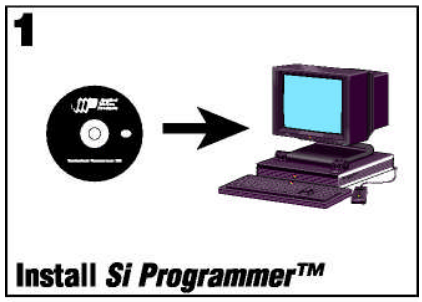
The Mis Programmer features include:
- Powerful, flexible, easy to use indexer.
- Nonvolatile program storage.
- Automatic, stand alone execution of stored program.
- Connection by a simple cable to your PC for programming (cable included).
- Programmable inputs & outputs for interacting with other equipment.
- Interrupt on input, branch on end of travel limit, branch on servo fault.
- Instructions for motion, triggering, branching, loops, subroutines, time delays and more.
- User defined units such as inches, degrees, gallons, etc.
- Optional man machine interface (MM I) allows operator to enter distances, speeds, loop counts and more.
- Built-in servo configuration and tuning including a digital storage scope.

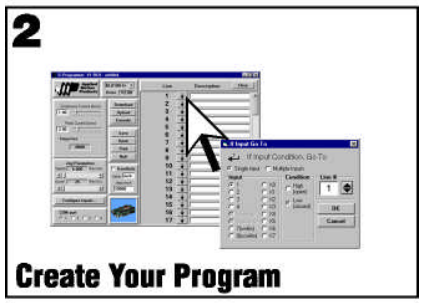
To operate your Mis product, you must do the following:
- Install our software program on your PC.
- Connect a motor (for the Mis-100, you'll need a motor and a pulse & direction drive).
- Connect any inputs or outputs that you require.
- Plug into your personal computer for programming.
- Connect & apply power.

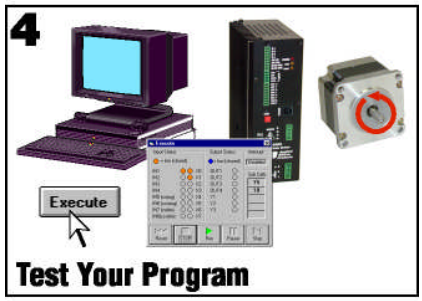
If you have trouble getting your Mis Indexer to meet your expectations, or if you want to suggest improvements to the product or this manual, give us a call at (8621) 52634688 or write to support@moons.com.cn.

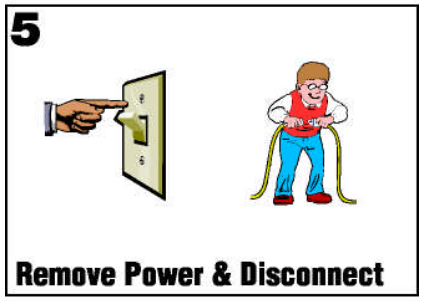
*Note: This manual was prepared for Mis Programmer 2.5 and the latest drive firmware. If your drive contains a previous firmware revision, then some of the features described in this manual may not be available to you. The Mis Programmer software wil alert you to this fact if you try to download a program that your drive cannot execute.*

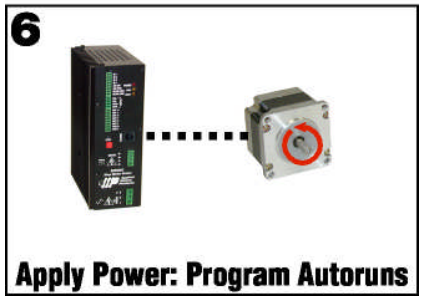
***If your PC contains an older version of the Mis Programmer software, please instal the new version. You do NOT have to uninstal the old version first. The latest Mis Programmer software is designed work with every version and model Mis drive ever produced.***





## Installing the Programming Software

The Mis Programmer software comes on a CD along with the necessary manuals and other software. Before you can use the software, you must install it on your hard drive.

To run the Mis Programmer software, you must have a computer with the following requirements:
- IBM compatible 386, 486, Pentium or better CPU. Pentium recommended for best performance.
- Microsoft Windows 95, 98, 2000, ME, NT or XP.
- At least 32 MB memory.
- 20 MB available hard drive space.
- VGA monitor, or better. 16 bit color setting recommended (65,535 colors, sometimes called High Color)
- Mouse or other input device
- CDROM or 3.5" floppy disk drive
- A nine pin serial port must be available, preferably COM1. If you don't have a serial port, get a "USB Serial Adapter."

The software installation is highly automated, like most Windows programs, so the process is simple:
- Put the CD into your CDROM drive.
- A CD Browser should appear within 60 seconds.
- Select "Install Software".
- From the next screen, select "Mis Programmer".

**If the CD does not autostart** the autoplay feature may have been disabled on your PC. This will continue to cause problems for you until you turn Autoplay back on. Please consult Help under your Windows Start menu or contact the manufacturer of your PC. If you are in a hurry (and who isn't?) you can manually launch the CD by double-clicking the My Computer icon. Then right click the Mis CD icon and choose Autoplay.

**If your PC does not have a CDROM drive**, but you have access to one that does, you can make your own diskettes, as follows:
- Insert the CD into a computer that has both a CD and a 3.5" floppy disk drive.
- Wait for the CD Browser.
- Choose "Browse CD Contents".
- Locate the folder "Mis Prog Disks".
- Inside this folder will be other folders, called "Disk 1", "Disk 2", etc.
- Open the folder Disk 1. Copy the files to a blank diskette and label it "Mis Programmer Disk 1".
- Repeat for the other disks.
- When you are ready to install from the diskettes, insert Disk 1 and choose Run from the Start menu. Type the command "A:\setup". The automatic installer will take over from there.

You can also download our software and manuals from www.moons.com.cn.

If you are unable to make your own diskettes, you may call the factory and request a set.

**If you encounter errors during installation**, it is usually due to a lack of memory or conflicts with other programs that are already running on your PC. If you experience an error while installing the programming software, quit all other Windows applications and try again. Holding down the ALT key and pressing TAB will show you all the programs currently running on your PC. Laptop computers generally present the biggest challenge to installation, as they often come pre-loaded with programs that automatically execute on start-up such as Microsoft Office and battery managers. Furthermore, laptops usually have the least memory.

***Programming Notes:***
***1. Always apply power to Mis hardware after the Mis Programmer software is running on your PC.***
***2. When downloading to the indexer/drive, make sure the JOG inputs are not activated. If in doubt, remove the JOG CW and JOG CCW connector plug.***

If all else fails, restart your computer and press the F8 key when you see the message “Starting Windows...” Windows will start in “Safe Mode” which generally solves the installation problem. Once you have successfully installed the software, you can reboot and operate Windows normally.

The programming software will install more easily and run much faster if you have more memory.

Several example programs are instaled with your programming software. It’s a good idea to load some of the examples and look at them; they may help you with your own application.

### Display Settings

The Mis Programmer works well with any display resolution. At 640 x 480, the Mis Programmer will exactly fill your screen. At higher resolutions, like 800 x 600 or 1024 x 768, there will be room left over on the screen for other applications, or to expand the Mis Programmer window so you can see more program lines. 16 bit color setting, or higher, is recommended.

***In formation in the program window wil not display correctly if your display is set for “Large Fonts.”*** Please use the “Small Fonts” setting when running the Mis Programmer software. The display settings are found under “Start…Settings.. .Control Panels”.

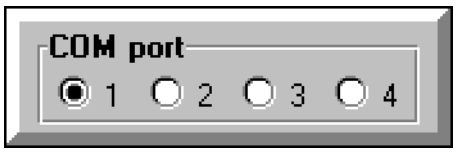
## *Connecting to your PC*

Locate your computer within 6 feet of the Mis hardware.

Your Mis product was shipped with a black adapter plug connected to a telephone line cord. Plug the large end into the COM1 serial port of your PC and the other end into your indexer. Secure the adapter with the screws on the sides. If the COM1 port on your PC is already used by something else, you may use the COM2 port for the Mis Indexer. On some PCs, COM2 will have a 25 pin connector that does not fit the black adapter plug. If this is the case, and you must use COM2, you will have to purchase a 25 to 9 pin serial adapter at your local computer store.

***Never connect the Mis Indexer to a telephone circuit. It uses the same connectors and cords as telephones and modems, but the voltages are not compatible.***

You may also need to set the COM port in the Windows software. When the software is loaded, it looks for the first available COM port, but doesn’t always find the one you’ve plugged into.




You can choose the port yourself by clicking on one of the “COM port” option buttons. If the port exists and is not already in use, the programming software will use it to communicate with the Mis Indexer.

### *What if my PC has no Serial Port?*

You can use a USB port as a serial port if you obtain a “USB Serial Converter”, which looks like a short cable. We like the Port Authority “USB Serial DB9 Adapter”, from CablesToGo.com, part number 26886. It’s inexpensive (usually under $30) and works great.

***Programming Notes:***
***1. Always apply power to Mis hardware after the Mis Programmer software is running on your PC.***
***2. When downloading to the indexer/drive, make sure the JOG inputs are not activated. If in doubt, remove the JOG CW and JOG CCW connector plug.***

# Programming

You may have noticed that the Mis Indexer doesn't have any switches or knobs on the front panel. There are also no jumpers inside. Just about everything you want the Mis to do is controlled by software. The Mis Programmer software that comes with the drive allows you to set the motor current, the step resolution, jogging parameters and limit switch polarity. It also helps you write complex motion control and machine interaction programs.

***Note: If you are using a BL or BLU servo drive and it hasn't been configured and tuned yet, you should skip ahead to the "Servo Tuning" section, and come back here when you're done.***

The Mis indexer has a user program capacity of 100 lines. In this space, you can design one or more motion and machine control programs. More than 20 commands, or instructions, are available for this purpose.

Six of the instructions involve pure motion: *Feed to Length*, *Feed & Set Output* and *Feed & Return* are fixed distance moves. *Feed to Position* is a move to an absolute position. *Feed to Sensor* and *Feed to Sensor & Return* move relative to a sensor that is wired to one of the inputs. *Seek Home* searches for a home sensor, "bouncing off" the limits if necessary to find it.

Two instructions handle timing: *Wait Time*, which causes your program to stop for a specified amount of time. Wait Input waits for one of the inputs to reach a specified state before continuing the program.

Five instructions control program flow. *Go To* makes the program jump to a particular line. *If Input* jumps to a line if one of the inputs meets a specified condition, otherwise, the program just goes on to the next line. *Repeat* and *End Repeat* set up a loop wherein you can repeat the same instructions many times. If your program terminates a Repeat loop before it's finished (using an If Input instruction) you can reset the loop count with a *Reset Repeat Loop* instruction.

One instruction, *Set Output*, allows you to signal other equipment that you have reached a particular place in your program.

Using the *MMI Prompt* instruction with the optional MMI (man-machine interface, or operator panel), the opera-tor can enter distances, speeds and repeat loop counts on a keypad. The drive can also display messages for the operator, pause the program until the operator presses the ENTER button, or ask the user to make a decision and respond by pressing the YES key or NO key.

A *Comment* instruction allows you to leave notes in your program so that it's easier to understand.

*Set Absolute Position* lets you define the present motor position in absolute terms.

*Change Current* gives you more control over the motor current - turning the current off, resuming the previous level, or defining a new current setting - anywhere in the program.

*Hand Wheel* lets the user position the motor and load precisely using a CNC hand wheel.

By combining the instructions in different ways, you can construct a nearly infinite variety of useful programs and motion profiles. Before entering your program, you'll want to spend a little time thinking about how to accomplish your objective. Then, once you have a clear idea of what to do, you can begin entering the instructions and parameters.

***Note: some programming features wil not be available if you are using an older Mis drive because the drive firmware doesn't support them.***

## Which Software Version do I Have?

There are actually two software programs associated with the Mis Indexer. The first is the Mis Programmer Windows program that you installed on your PC from the floppy disks. After you double click on the icon and program begins to load, you'll see a picture of a lightning storm. The software version is displayed at that time. After the pro-gram is loaded, you can click on the Applied Motion Products logo to see the software version, our internet address and our phone and fax numbers.

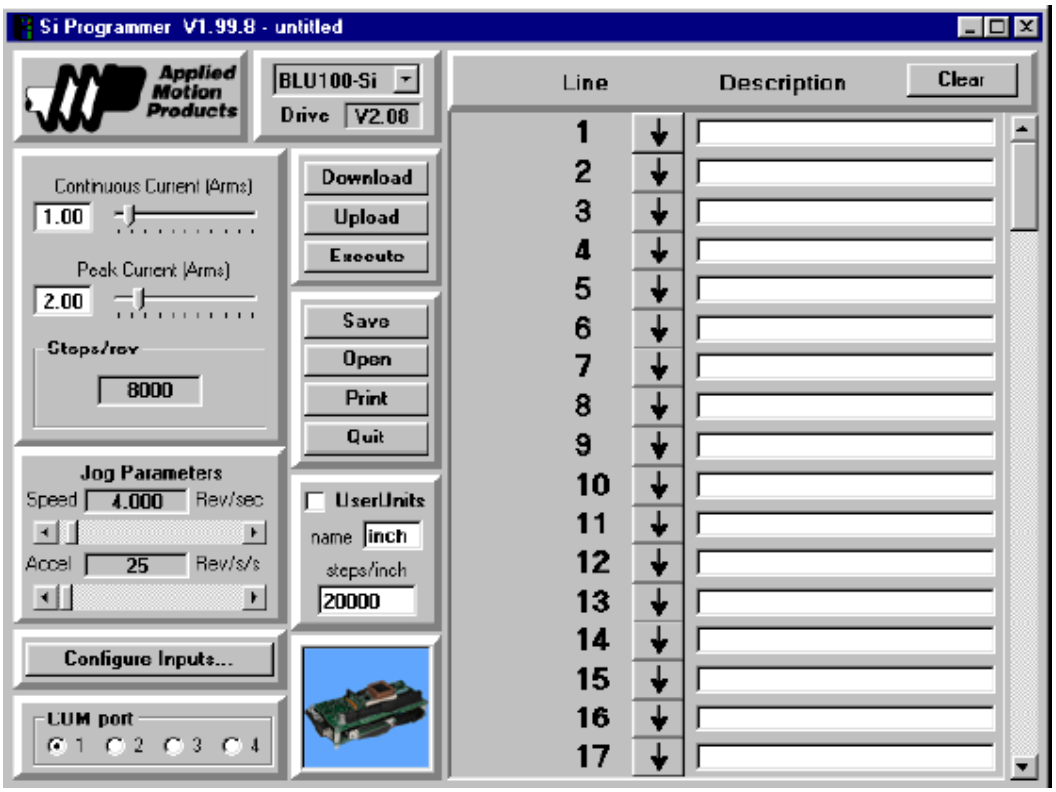
A second software program resides in a chip inside the Mis Indexer. Since software in a chip is usually called "firmware", we will refer to it as firmware for the rest of this manual. It is the Mis Indexer firmware that runs your drive and executes the program you've downloaded. When you connect your drive to the PC and turn the drive on, the drive firmware version is displayed near the top of the screen.
If you are using a BLU servo or STAC6 drive, then an additional firmware program resides in the Digital Signal Processor on the drive's main circuit board. To discover the firmware version of the DSP, click on the Mis firm-ware version shown on the main window.

If you call your distributor or Applied Motion Products for support, we will want to know the versions of both the Mis Programmer and the drive firmware, so please write them down before calling.

You may have noticed the list box just above the firmware version number. The Mis software is designed for programming all of our Mis drives. Please select the appropriate drive from this list before you begin programming.

## Entering Your Program

If you installed the software back in the section entitled "Installing the Programming Software" then you're ready to go. If not, please go back and review that section.



To activate the software, click on the Start button, then Programs. ..Applied Motion Products. ..Mis Programmer. The main programming window will soon appear, as shown on the previous page. The title bar will display the Mis Programmer software version.

If you have an Mis Indexer connected to the PC, turn it on now. After you apply power, your computer should beep. The “Drive” box will display the version number of the Mis Indexer firmware that’s in your drive.

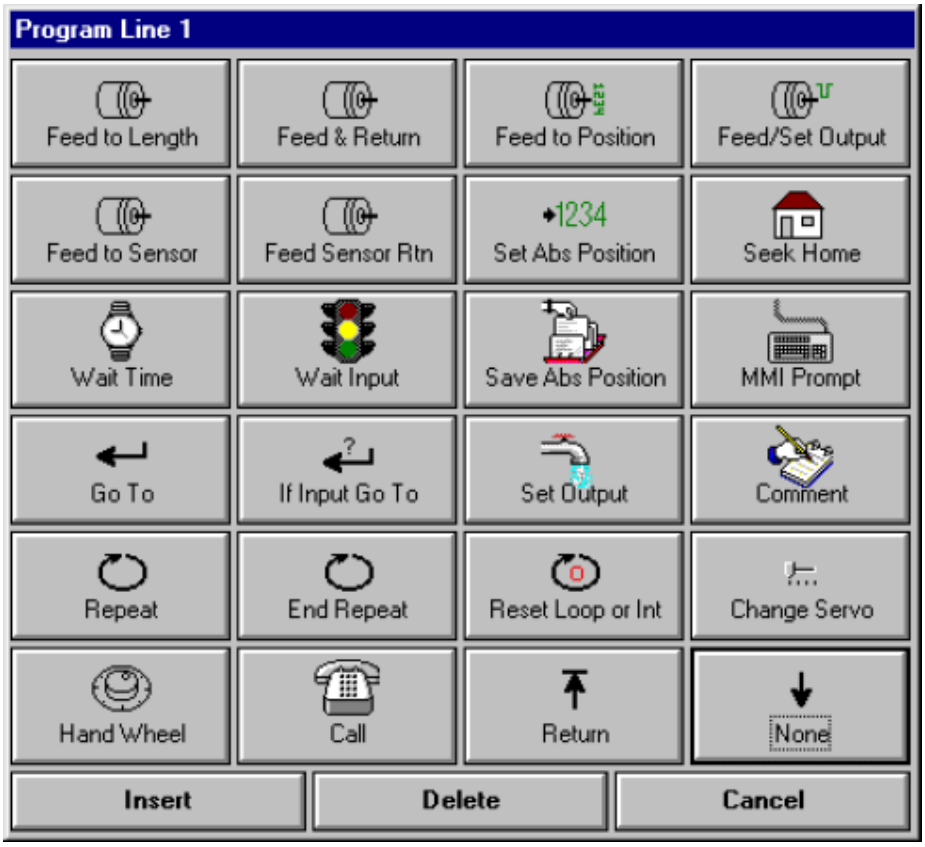
If you don’t have any Mis hardware connected to your PC, you can still write programs. Before you begin, you should select the appropriate Mis device (Mis5580, 7080i, etc) from the list box above the word “drive.” That way you will have access to the specific features of the hardware you plan to use.
Let’s enter a simple program. We’ll start with a simple time delay, by putting a Wait Time instruction on the first line.

Next to the large number 1 in the Program Window is a button showing the “nothing” icon. That indicates that there is no instruction for that line. Anytime the Mis Indexer encounters a “nothing” program line, it simply moves on to the next line, as the icon implies with a downward pointing arrow. After the Mis Indexer executes the instruction on line 100, it automatically jumps to line 1, unless the instruction on line 100 makes it jump somewhere else.

To enter an instruction on program line 1, click once on the program icon. You should now see the “Program Menu”, shown below.

Click on the button marked “Wait Time”. The Wait Time dialog box will appear. Enter 1 second in the text box and press OK.

The first line of your program should now display the Wait Time icon, and the description “Wait 1 second.”




Click on the icon button for pro-gram line 2. This time when you see the “Program Line...” box, click on Feed to Length. In the Feed to Length box, enter the distance as 20000, then slide the speed bar to 10 rev/sec. Click OK.

The second program line should now show the motor icon and the caption “CW 20000 steps, 10 rps.”

Click on the line 3 icon. Choose “Go To”. When you see the Go To dialog, the line number will already be set to 1. Click OK. Your program should now resemble the following:

If you have a motor connected to your drive, you can test the program now.

Set the current to match your motor’s rated current. Then, click on the Download button near the middle of the screen. If your drive is on, and is connected properly, the download box will appear and show you the progress of the download, which takes 1-3 seconds. (The transfer time is governed by the speed at which the Mis Indexer can rewrite its internal, nonvolatile memory, and by the size of your program.) Once the download has been completed, you are ready to execute the program.

Press the Execute button. You’ll see the execute box in the middle of the screen. (If your indexer/drive has firmware prior to 1.40, you’l see a simpler execute box than the one shown here. Older drives are not able to send real time status information to the PC, and cannot respond to advanced commands like Pause and Single Step.)

Every second, the motor should move one revolution. (Assuming that you have left the resolution setting at the default 20,000 steps/rev. If you have a servo drive, then 20,000 steps is probably more than one revolution.)

Not so exciting, perhaps, but you have to admit it was easy to do.

More complex programs are entered in the same manner, you just enter more lines and you’ll be more concerned about the exact parameters and their importance in your application. We’ve designed the Mis Indexer to be easy and fun to use. If you can think of anything we’ve forgotten, please give us a call or send a fax. We continuously improve our products, and are always developing new ones based on what we’ve learned from you, our customer.

*Note: if you need to edit an instruction that’s already in your program, and wish to go directly to the appropriate dialog box, hold down the shift key while you click on the instruction icon.*

## *Copying Instructions*

There may be occasions where you want to make an exact copy of an instruction, or perhaps a copy with only one or two parameter changes. The fastest way to copy an instruction from one line to another is to point the mouse at the instruction icon that you want to copy, and drag the icon onto another one elsewhere in the program. This is referred to as “Drag and Drop.”

For example, say you’ve entered a *Set Output* instruction on line 5 of your program, and you’d like an identical *Set Output* on line 11. Position the mouse over the *Set Output* icon on line 5, then click and hold the mouse button. Move the mouse until the icon is over the line 11 icon. Let go of the button, and the instruction with all of its parameters will be copied to line 11.

## *Inserting and Deleting Program Steps*

**Insert New Step**
The time will no doubt come when you've entered many program lines only to realize that you need to add an instruction right in the middle. We could be cruel and tell you that you'll have to reenter most of the instructions to make room for the new one. But, in the spirit of making the Mis Indexer easy to use, we've included a command to insert a new instruction anywhere in your program.

It's easy to do: just click on the program icon where you want the new instruction to go. You'll get the Program Line... dialog, as usual. Instead of choosing one of the 20 instructions, click on the command button marked "Insert." The instructions are reordered, and the line you need for your new program line is available.

You can then click on the open line and select an instruction as you normally would.

**Delete Step**
In addition to inserting a line in your program, you can delete one to make room for others farther down. You may, for example, have some available space in the middle of the program, but want to add an instruction near the end.

Click on a program line that you don't need. When the Program Line... dialog comes up, select "Delete." The line you selected for deletion will be gone, and all the other lines will move up one position, leaving a blank spot at the end of the program.

By combining insertions and deletions, you can place your program lines wherever and whenever you need to.

## *User Defined Units*

The Mis Programmer normally expects you to enter distances in steps and speeds in revolutions/sec. However, you can also define your own units.

To define your own unit, first enter a name in the name box. In the example at the left, the name is "inch." The name you enter cannot be longer than four characters.

Next, you must enter the number of motor steps in one of your units. For example, 20,000 steps/rev with a screw pitch of 2 turns per inch is

$$(2revs/inch)(20{,}000steps/rev) = 40{,}000steps/inch$$

Sometimes screw pitches are expressed as inches per turn (0.2 for example.) If that case, your steps/inch would be the steps/rev divided by the inches/turn.

Be sure to check the "User Units" check box. You will notice the units of the jog panel change from "rev/sec" to "inch/sec." Any program instruction dialog you open will also be operating in inches and inch/sec.

Other choices of steps/rev can be useful when defining units. If you have an English screw and wish to work

in millimeters, for example, setting the steps/rev to 25400 is helpful. Then the user unit definition for the 2 turn screw would be

$$\frac{(2revs/inch)(254{,}000steps/rev)}{25.4mm/inch} = 2000steps/mm$$

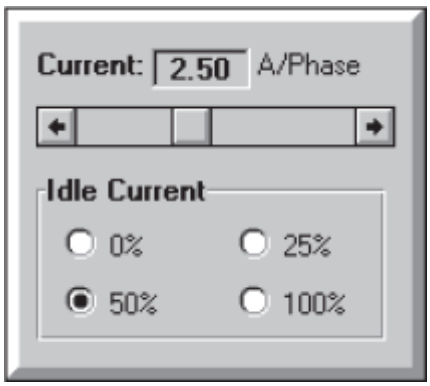
***Note: If the Mis hardware you are programming has firmware prior to version 1.28, you can stil pro-gram it with user defined units, but when you download your program, the indexer wil not remember your unit name and pitch. The program wil execute correctly, but if you upload, the user unit definition wil not upload with the rest of the program.***

## Front Panel STOP Button

Some Mis products have a red button on the front panel marked “STOP.” This button can be used to interrupt motion at any time. After pressing the STOP button, the motor will stop and the front panel Power LED will then flash until the AC power is removed. If the indexer/drive is connected to a PC running the Mis Programmer, the software will alert the user on screen to the condition, and ask if you want to reset the indexer/drive from the PC.

## Setting the Step Motor Current

The drive current must be set to match the motor. First, determine the rated current for the motor. If you are using one of the motors recommended in the User’s Manual for your Mis indexer/drive, the User’s Manual lists the rated current. Otherwise, you’ll have to go by the manufacturer’s rated current, which is usually printed on the motor label. You can operate a motor at less than the rated current. It will have less torque than it would at the rated current, but will run cooler and make less audible noise.




Depending on how you connect the motor to the drive, the current setting on the drive may differ from the rated current of the motor. For example, if you’re using an Applied Motion motor, follow these rules:

**Four lead motor:** Use the rated current. The motor can only be connected one way.

**Six lead motor:** the nameplate current is for the center to end connection. If you choose to connect the motor in series, divide the current by 1.4.

**Eight lead motor:** the nameplate current is for center to end. For parallel connections, multiply the current by 1.4. For series divide by 1.4.

In the Mis Programmer software, the current is controlled in the main window by the panel on the upper left side of the screen. To adjust the current setting, just slide the scroll bar left or right. Precise adjustments can be made by clicking on the arrows at each end of the scroll bar.

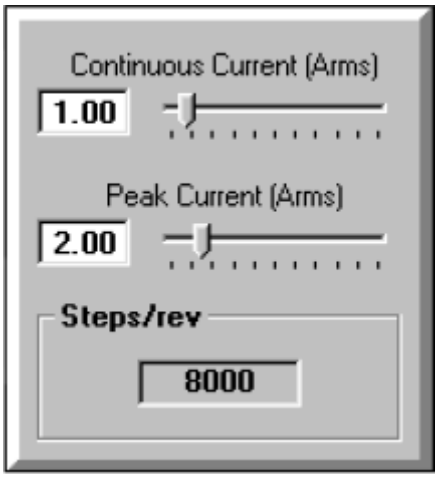
Note: current setting only applies to Mis indexers with built-in drives, like the Mis5580 and 7080i. The Mis-100 does not have an internal motor driver, so the current setting has no effect on the Mis-100.

### *Idle Current Reduction*

Your drive is equipped with a feature that automatically reduces the motor current anytime the motor is not moving. This reduces motor and drive heating. For example, setting the idle current to 50% reduces drive heating by about 50% and lowers motor heating by 75%. This feature can be set at any of four levels: 0%, 25%, 50% and 100%. The 100% setting is useful when a high holding torque is required, as the drive does reduce the current at all. The 0% setting is for applications in which no holding torque is required.

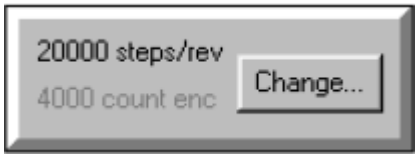
To minimize motor and drive heating we highly recommend that you use the idle current reduction feature unless your application strictly forbids it. The idle current setting is chosen using the option buttons in the motor current panel.

## *Setting the Servo Motor Current*

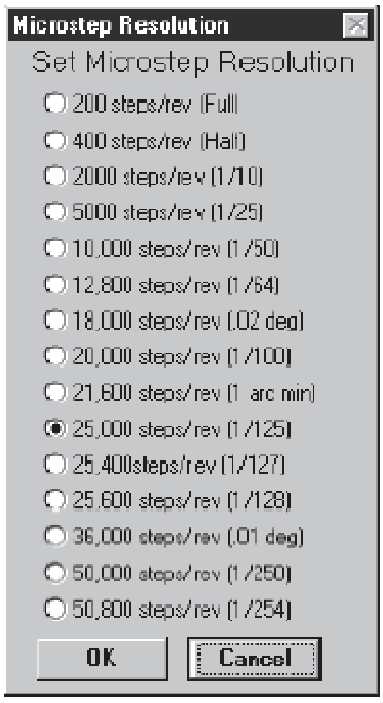



The drive current must be set to match the motor. First, determine the rated current for the motor. If you are using one of the motors recommended in the User's Manual for your Mis indexer/drive, the User's Manual lists the rated current. Otherwise, you'll have to go by the manufacturer's rated current, which is usually printed on the motor label. You can operate a motor at less than the rated current. It will have less torque than it would at the rated current, but will run cooler and make less audible noise. BLU Servo Drives have separate settings for continuous and peak current. The peak current only lasts for a few seconds but allows you to get extra torque from the motor for higher acceleration. Normally you would set the peak current to 3X or 3 times the continuous current, unless there is a mechanical limitation (such as a gearhead). BL servo drives automatically set the peak current to 2X the continuous.

## *Microstepping (Step Motors Only)*




Most non-microstep step motor drives offer a choice between full step and half step resolutions. In full step mode, both motor phases are used all the time. Half stepping divides each step into two smaller steps by alternating between both phases on and one phase on.




Microstepping drives like the Mis5580 precisely control the amount of current in each phase at each step position as a means of electronically subdividing the steps even further. All Mis indexer-drives offer a choice of 13 step resolutions, starting at 2000 steps per revolution. The highest setting divides each full step into 254 microsteps, providing 50,800 steps per revolution when using a 1.8° motor.

Other Mis products may offer a different selection of resolutions than the ones shown here. The Mis Programmer automatically presents the resolutions that are available on your hardware if the model number of your indexer-drive is shown in the "drive box" near the top of the screen.

In addition to providing precise positioning and smooth motion, microstep drives can be used for motion conversion between different units. The 25,400 step/rev setting is provided as a means of converting motion from metric to English. (There are 25.4 mm per inch.) Other settings provide step angles that are decimal degrees (36,000 steps/rev makes the motor take 0.01° steps.) Some settings are used with lead screws. When the drive is set to 2000 steps/rev and used with a 5 pitch lead screw, you get .0001 inches/step.

The selection of microstep resolution can be important if you are using the optional MMI, and plan to scale the distances or speeds that the user enters.
If in doubt, choose 20000 steps/rev. The motor will run smoother and more quietly at 20000 steps/rev than at lower resolutions like 2000.

The microstep resolution of the Mis Indexer is set using the programming software. The resolution appears at all times in the Steps/revolution panel on the left side of the main window. To change the resolution, click on the Change button and you'll see a dialog box with option buttons for each resolution. Click the one you want, then click on the OK button.

If you change the step resolution and there are motion instructions in your program, the software will warn you that the distances may need to be changed (because they are in steps). It also offers to automatically scale the distances so that the distance in revolutions remains the same.

***Note: If you are using an Mis-100 indexer, you must also set the step resolution at the drive so that it matches the Mis Programmer setting.***

## *Jogging*

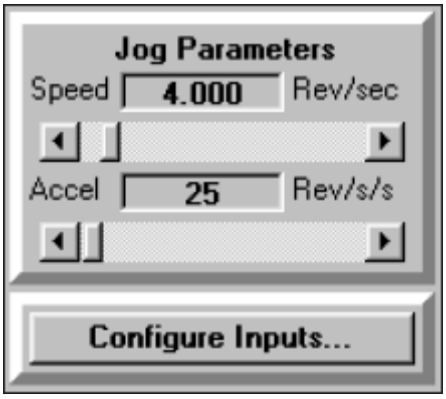
Two of the Mis Indexer input terminals are provided for jogging the motor.

If the Mis Indexer is connected to a PC with the programming software running, the jog inputs will function under two conditions:

- if the program is not executing (while connected to the PC)
- if the program is executing a Wait Input command.

If the Mis Indexer is operating in stand alone mode (i.e. without a computer attached) then the jog inputs work when the program is executing the Wait Input instruction.

To set the Jog Speed and Jog Accel/decel rate, adjust the scroll bars in the main programming window.




We recommend setting the accel/decel to a modest rate. 25 rev/s/s usually works well unless you have a very high inertial load, in which case you should set it to a lower rate. The range of jog accel is 1 to 3000 rev/s/s.

The range of jog speed is .025 to 50 rev/sec. The speed you choose will depend on your application.

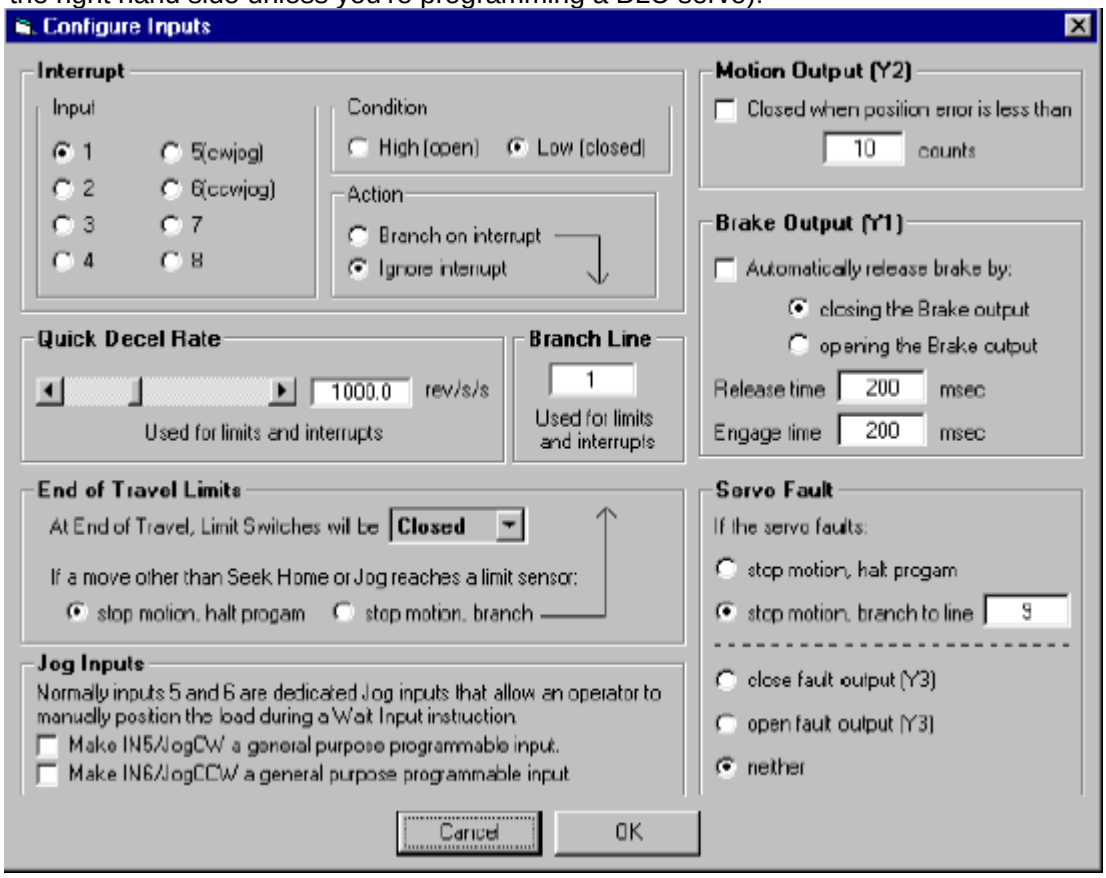
If you don't need to jog, you can use the jog inputs as input 5 and input 6, for use with Feed to Sensor, Wait Input and If Input instructions. The CW JOG input can be assigned as a general purpose input by checking the box marked "Make IN5/Jog CW a general purpose programmable input." The IN6/Jog CCW input can also be used as a general purpose programmable input. These settings are found in the Configure Inputs dialog.

If you have an Mis indexer/drive with firmware version 1.40 or later, you can use the arrow keys on the optional man machine interface (MMI) for jogging. This is an option in the Wait for MMI Enter instruction.

Firmware version 2.08 and later alow three choices of jog speed in the Wait Input and MMI Enter instructions: global jog speed (the one set by the main screen panel shown above), local jog speed (specific to each instruction) and MMI Jog Speed (uses an MMI speed variable to alow the user to enter a jog speed.)

## Configure In puts Dialog

On the main screen, just below the jog settings is the Configure Inputs button (it's labelled Configure I/O when you're programming a BLU servo drive). Clicking this button brings up the following dialog box (you won't see the right hand side unless you're programming a BLU servo):




### Interrupt

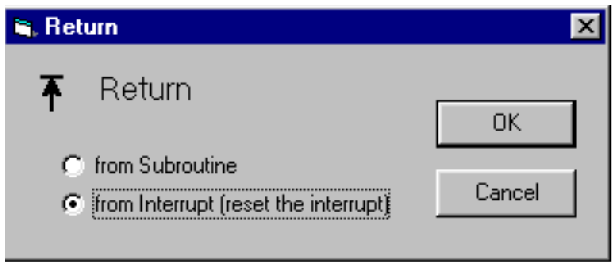
If your drive has firmware version 2.08 or later, you can assign one of the inputs to interrupt your program and branch no matter what your program is doing. For example, your Mis drive may be part of a larger material handling system. If something "downstream" of the Mis drive cannot handle more product (perhaps a conveyer jammed, a box is full and has not been removed or a labeler has run out of labels or ink), you may want to command the Mis drive to stop what it is doing and wait for the condition to clear.

Interrupts are set in the Configure Inputs dialog, as shown above.
The first step is easy: just choose an input and a condition for the interrupt. If a sensor closes when a fault occurs and that sensor is connected to input 3, choose "input 3" as the input and "low/closed" as the condition. Then select "Branch on interrupt" and choose a line number. That's where you will put instructions to handle the interrupt "recovery". Generally this means waiting for the interrupt condition to clear (a Wait Input instruction can do that), and then returning from the interrupt. This is where is gets a little tricky, so please read carefully.

**Two Types of Interrupts**
The Mis drives support two different kinds of interrupts: Call interrupts and Go To interrupts. You'll set the Configure Inputs dialog the same way for both. The distinction is in the way you build your "interrupt handler" (the instruction that you place at the interrupt branch address.)

**Call**

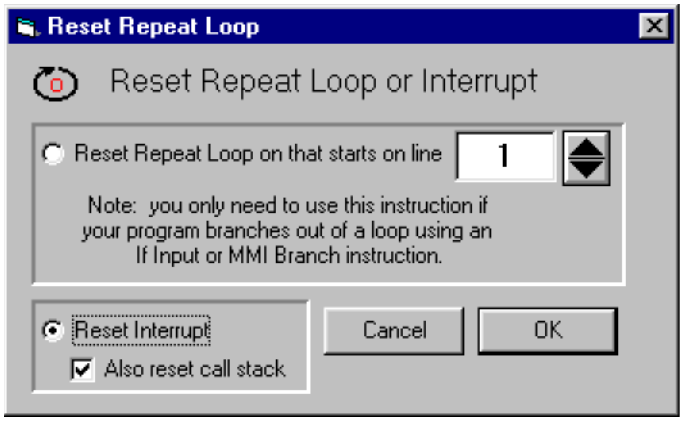
This type of interrupt uses the “Call/Return” mechanism to allow your program to return to whatever it was doing before the interrupt occurred. Sometimes this is referred to as “slide hold” because you can interrupt a move, then resume the
move after the interrupt condition has gone away. Here’s how it works:




1. When the interrupt input matches the condition you specified, the drive stops whatever instruction it was executing, including any move.
2. The drive pushes a return address onto the Call Stack. If you don’t know what the Call Stack is, please read the section entitled “Call and Return.”
3. The program branches to your interrupt address. The interrupt is also disabled so that you don’t keep branching to same address over and over again, with your program unable to do anything else. If your drive is still connected to your PC with the Mis Programmer software running (we call this Development Mode), you will see the interrupt status change from “Enabled” to Disabled”.
4. You should wait for the interrupt condition to clear. If you are using an MMI, you may want to alert the operator.
5. Once the condition has cleared, use the Return instruction to resume your main program. Be sure to choose the option “Return from interrupt”, not “return from Subroutine.” This instruction will enable the interrupt and return to the instruction that was running when the interrupt first occurred. If this instruction is a Feed to Sensor, Feed to Position or Seek Home move, the move will usually reach its intended target. Feed to Length moves generally overshoot under these conditions because after the interrupt, the move starts over from the beginning. For example, if a 6 inch Feed to Length move was interrupted after going 3 inches, the total move distance will be 9 inches.
6. For a Call interrupt, do NOT use the Reset Interrupt as part of your interrupt handler or you will experience a stack underflow. Also, you don’t want the interrupt enabled until you return to the main program or the interrupt handler could be interrupted causing a big stack mess (BSM).

**Go To**

In many applications, it is better that the program not “pick up where it left off” after an interrupt. You may want to start your program over from the beginning, or you may wish to always resume from some other spot. Here’s what you do:


Reset Repeat Loop
Reset Repeat Loop or Interrupt
Reset Repeat Loop on that starts on line
1
Note: you only need to use this instruction if your program branches out of a loop using an If Input or MMI Branch instruction.
Reset Interrupt
Also reset call stack
Cancel
OK

1. When the interrupt input matches the condition you specified, the drive stops whatever instruction it was executing, including any move.
2. The drive pushes a return address onto the Call Stack, which we will not actually use.
3. The program branches to your interrupt address. The interrupt is also disabled so that you don’t keep branching to same address over and over again, with your program unable to do anything else. If your drive is still connected to your PC with the Mis Programmer software running (we call this Development Mode), you will see the interrupt status change from “Enabled” to Disabled”.
4. You should wait for the interrupt condition to clear. If you are using an MMI, you may want to alert the operator.
5. Once the condition has cleared, you need to turn the interrupt back on. The Reset instruction can do that. Be sure to choose the “Reset Interrupt” option. If you forget this instruction, the interrupt will only work once. Be sure to choose the “also reset call stack” option. That will remove the interrupt address from the call stack and prevent a stack overflow.
6. Use a Go To instruction to complete your interrupt handler.

**Combining Subroutines and Interrupts**
The Call Stack is five levels deep. The interrupt always uses one level, so if you are using both subroutines and interrupts in your program, be sure to use no more than four stack levels for your subroutines. That way even if you are "four levels deep" on your subroutine calls (a subroutine calling a subroutine which calls a subroutine) you will still have one stack level free for the interrupt.

## *Quick Decel Rate*

When something unexpected happens you usually want to stop the motor quickly. The Quick Decel rate is used when a move is interrupted by an end of travel limit and when an interrupt becomes active. It is also used for servo faults and when you click the Stop button on the Mis Programmer's execution status box. If you set the Quick Decel rate too high, a step motor could "break lose" and coast to a stop, though that usually won't hap-pen unless you have a high inertia load. Servos are so responsive that a very high decel rate could cause damage to the mechanics of your system.

## *Limit Switches*

The Configure Inputs dialog of the Mis Programmer software contains a panel for selecting the type of limit switches or sensors that you have.

If your switches will close when the motor reaches a limit, select the option marked "closed." This is often referred to as a normaly open switch. If your switches are closed when the motor is not at a limit, and will open when a limit is reached, select "open." This type of switch is frequently called normaly closed. If you're not using limit switches in your application, you can select "not used," making the cw and ccw limit inputs available as inputs 7 and 8 for Wait Input and Feed to Sensor instructions. The limit inputs are always available for If Input instructions.

Earlier firmware versions stop the motor instantly with no decel ramp and always halt the program. The red Power LED on the front panel will flash, and no future motion is possible. You must remove power from the drive to reset this condition.

If the drive is connected to a PC, the programming software will alert the user to this condition and ask you if you want to reset the drive from the PC (instead of removing power).

### *What Happens When You Hit a Limit Switch?*

If you are **jogging** (using the JOG CW or JOG CCW inputs or the MMI arrow keys) and you hit a limit, motion will be disabled in the direction that you were traveling. You can then jog in the reverse direction to back away from the limit.

During a Seek Home instruction, the motor will reverse direction when a limit is encountered, and continue seeking the home sensor.

**If you encounter a limit during a Feed to Length, Feed & Set Output, Feed to Position, Feed to Sensor, Feed & Return or Feed to Sensor & Return** move, the Mis Indexer will immediately stop the motor.

**If you are using drive firmware version 2.08 or later**, the "Quick Decel rate from the Configure Inputs dialog will be used to decelerate the motor. What happens next depends on other settings of the Configure Inputs dialog. You can choose to halt the program. If so, your program and all motion will be stopped. The drive will flash the power LED and will perform no further action until power is cycled.
If you choose "stop motion, branch" in the Configure Inputs dialog, the motor will stop and the program will branch to the line you specify. You can put an "error recovery routine" at the branch line number to handle the limit problem.

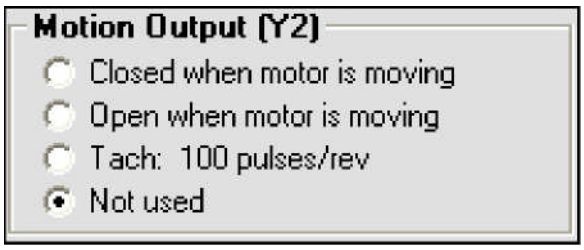
**Interrupt and Limit Branch**

If you have chosen “branch on interrupt” and “branch on limit” in the Configure Inputs dialog, both events will branch to the same program line. You may need to distinguish between the two events. If you are using the Call/Return style of interrupt, then you must have a Return from Interrupt instruction in your interrupt handler. But you must not allow the limit branch to encounter that instruction or you will underf low the call stack. You can use an If Input instruction to detect the limit and branch to a different set of instructions. If your limit condition is “low” at the limit, then use “If Input 7 or 8 low”. If your limit condition is “high” at the limit, then use “If Input 7 or 8 high” to detect the limit in your handler.

### *Motion Output (BLU Servo)*

If you select this function, output Y2 will automatically close when the motor is at or near its target position. You define the meaning of “near” by entering a tolerance in encoder counts. The target position changes from the present position to the “end of move” position as soon as a new move begins. The motion output will not close in the middle of a move even if the servo position error is very low. The motion output is designed to tell an-other piece of equipment that you’ve “arrived.”

### *Motion Output (STAC6)*




The STAC6-Mis offers four options for the function of output Y2. Choose “closed” is you want Y2 to be closed when the motor is moving and open at all other times. “Open” causes Y2 to be open when the motor is moving and closed at rest. “Tach” is useful if you want another piece of equipment, such as a PLC with a built-in counter, to know how fast the motor is moving. When the “tach” option is chosen, Y2 produces 100 pulses per motor rotation. For example, if the motor is rotating at 5.5 revs/ second the tach frequency will be 550 Hz. Select “not used” if you want to control Y2 in your program by using Set Output instructions.

### *Brake Output (BLU Servo and STAC6)*

When power is removed from a servo drive (or when a fault occurs) the motor loses all torque. The load is then free to move if acted upon by an external force such as gravity or a spring. In applications where it is essential that the load not move when the servo is turned off, a brake is required. Sometimes the brake is added externally to the shaft or load; other times the brake in built into the motor. In either case, the brake holds position when power is removed and must be supplied with power to release and allow motion. A small amount of time is required for the brake to engage or release.

The BLU servo or STAC6 drive can automatically control the brake by wiring a power supply and a power relay in series with the brake coil. Another power supply, usually 24 volts DC, is then wired in series with the Brake output (Y1) and the relay input coil. Please consult the drive hardware manual for wiring details.

If you want the drive to automatically release the brake when it powers up and to engage it when power fails and/or when the motor is disabled, just check the Brake Output box. You must allow specify the “release time,” or how long you want the drive to wait before making any moves once it has sent the release signal to the brake. You must also tell the drive how long to wait for the brake to engage before turning off the motor.

***Please note that in case of a servo fault, the motor must be disabled immediately. The brake engage time is ignored and some load slippage may occur.***

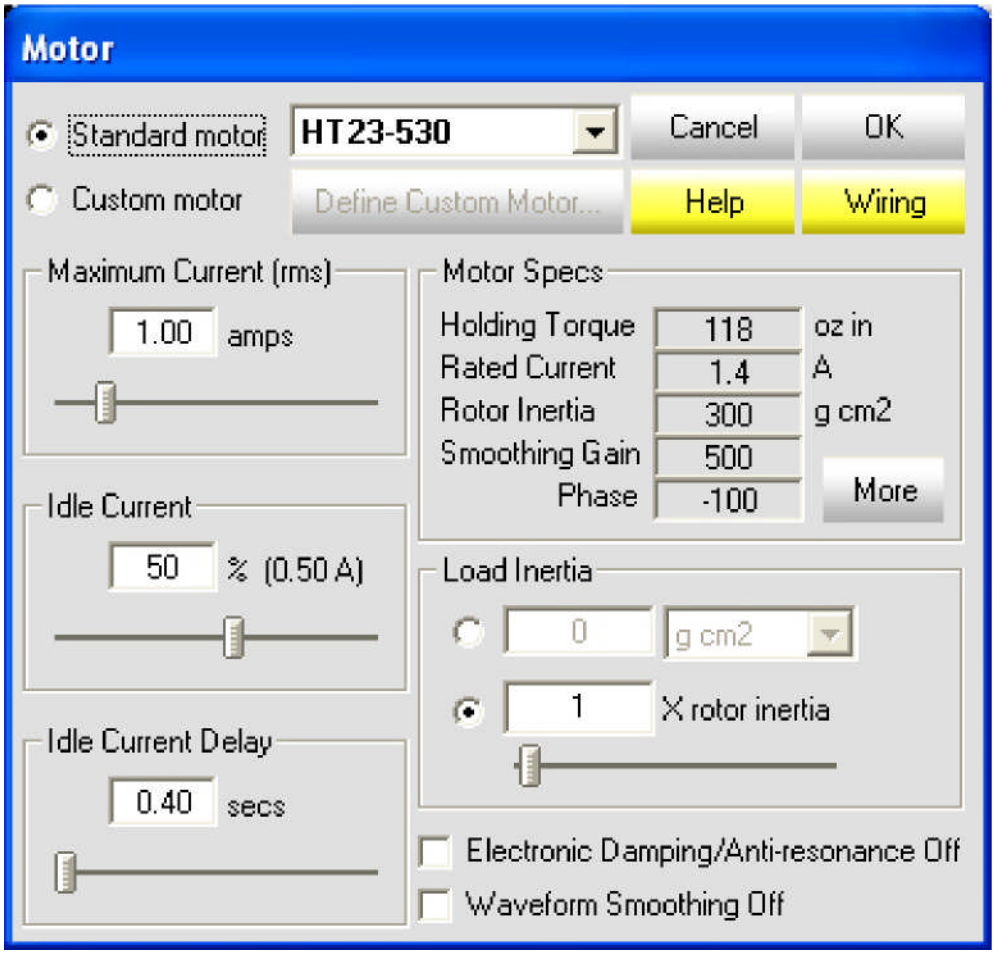
## Special STAC6 Dialogs

When a STAC6-Mis drive is selected from the drop down list, four special command buttons appear: Motor, Encoder, Alarms and Regen. Each button invokes a special dialog box for setting features unique to the STAC6.

The Encoder dialog is explained in the section “Encoder: STAC6”.

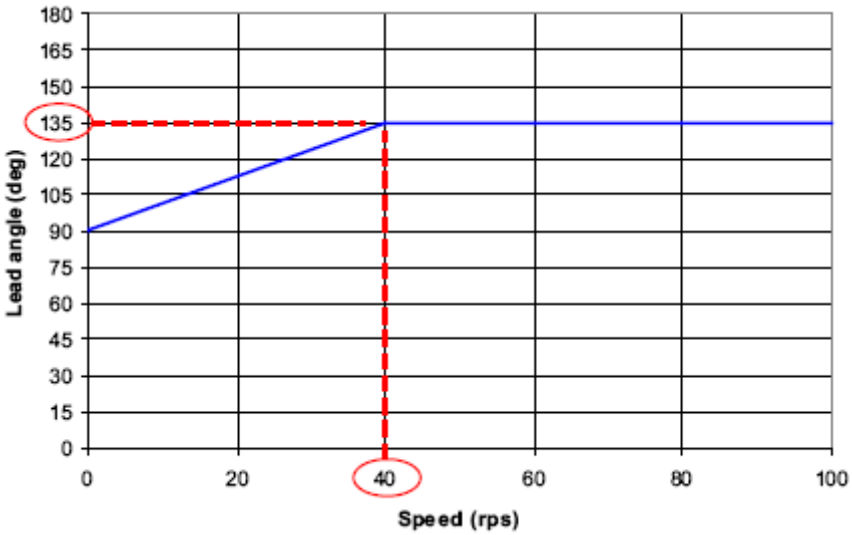
The **Motor dialog** is used to select a motor from the list of Applied Motion Products motors that are de-signed for use with the drive. For optimal performance of the antiresonance and electronic damping features, you must also enter (or estimate) the load inertia.

We can't stress enough the wisdom in using one of the recommended motors. We're not just trying to make money here, we want your application to be successful and the odds of that are highest when you have a high quality motor whose torque, rotor inertia and harmonic waveform content are precisely known. Furthermore, our motors include shielded cables to reduce electrical emissions and enhance safety and come with prewired mating connectors which further reduces the risk of error.
Having stated our case, if you still insist on using a different motor, it is possible. First you'll need some detailed information from the manufac
turer, including electrical specification (holding torque, rated current and rotor inertia) plus a wiring diagram. And make sure the motor is constructed from high quality magnetic materials that are suitable for operation with 160 volt busses such as that of the STAC6.



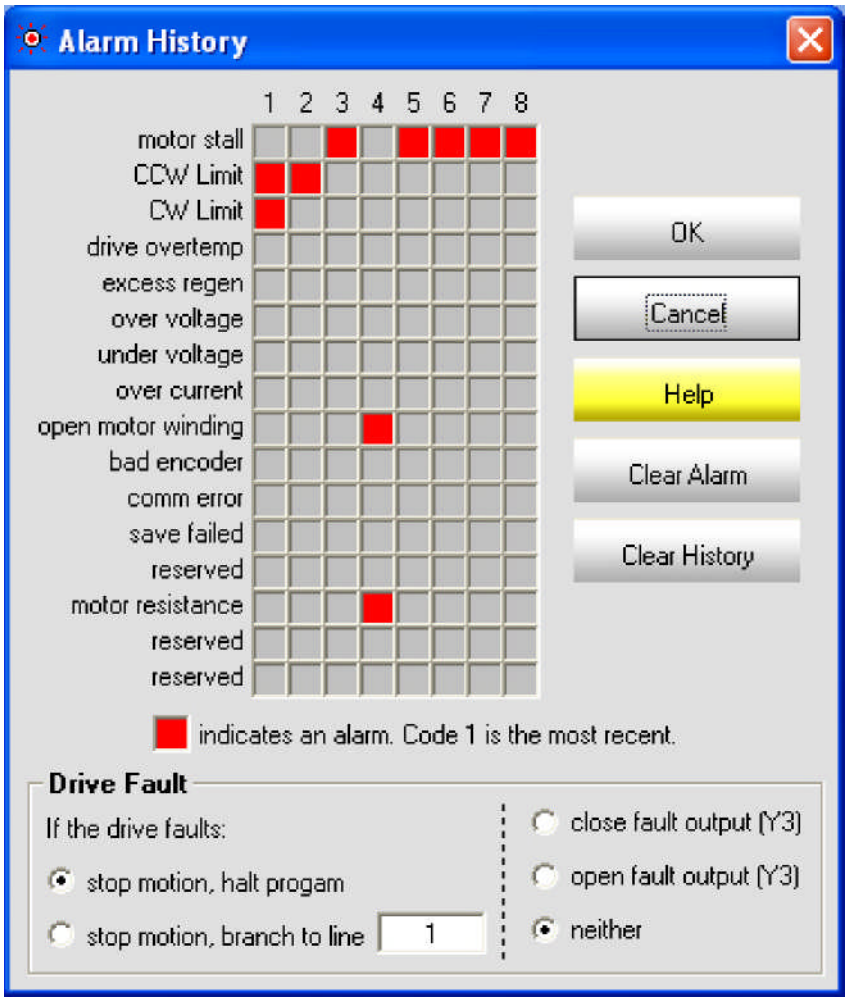

With this information in hand, choose the "custom motor" option and click on the Define Custom Motor button. You can enter the current, torque and inertia values into the Add New Motor dialog. For best smoothness of motion, you'll want to enter the harmonic distortion gain and phase. You may need to experiment by running the motor at a slow speed (typically 1 rev/sec) with different gain and phase values to see what works best. If you are at a loss for this data, set the gain and phase at 0.

If you plan to use the Encoder Stall Prevention feature (see the Encoder Dialog), you'll need to enter the maximum lead angle and the speed at which this "timing advance" peaks so that the STAC6 knows when it is producing maximum torque. A typical motor produces maximum torque at low speeds with a 90° lead angle. To produce maximum torque at higher speeds, the lead angle must be increased because of inductance and back emf. Above a certain speed, further increases in lead angle produce no benefit, so you must tell the STAC6 when to stop advancing the timing. In the example below, the lead angle is increased steadily from 90° at low speeds to 135° at 40 rev/sec, so you would enter "135 degrees at 40 rev/sec", as shown above. If you are not using Stall Prevention, these values are not needed by the STAC6.

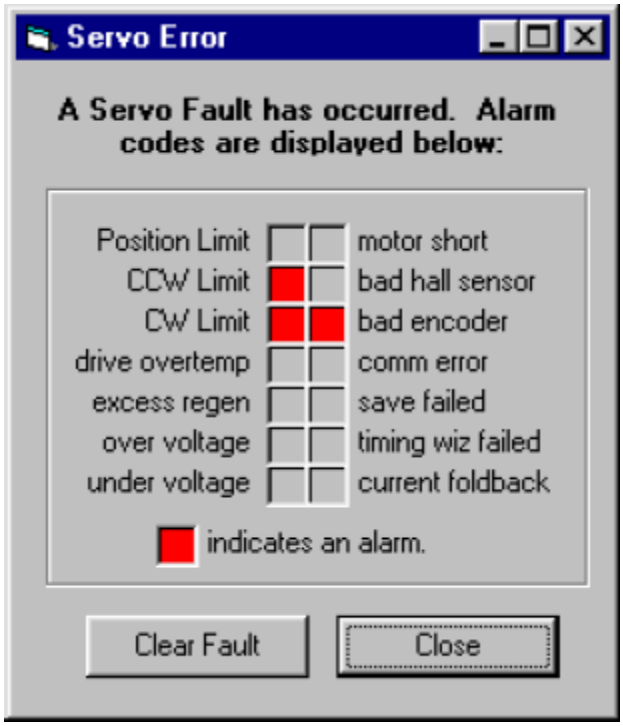



The Alarms dialog shows you the a chart of the eight most recent alarm conditions. It also allows you to clear an alarm or to clear the entire history.

The alarms dialog also allows you to dedicate output Y3 to the task of signalling a fault condition to other electronic devices. You can also choose to branch to a particular branch of your program should a fault occur.



## Servo Faults

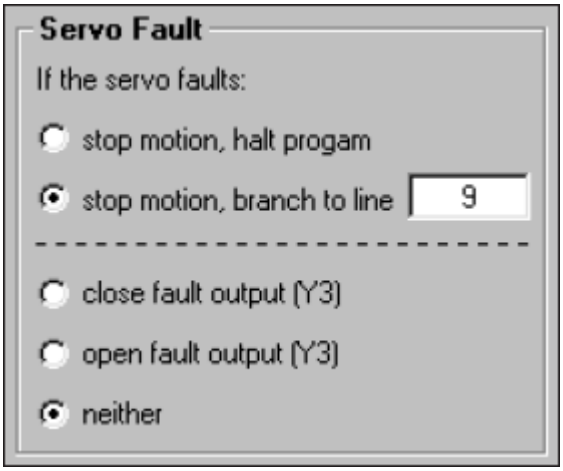


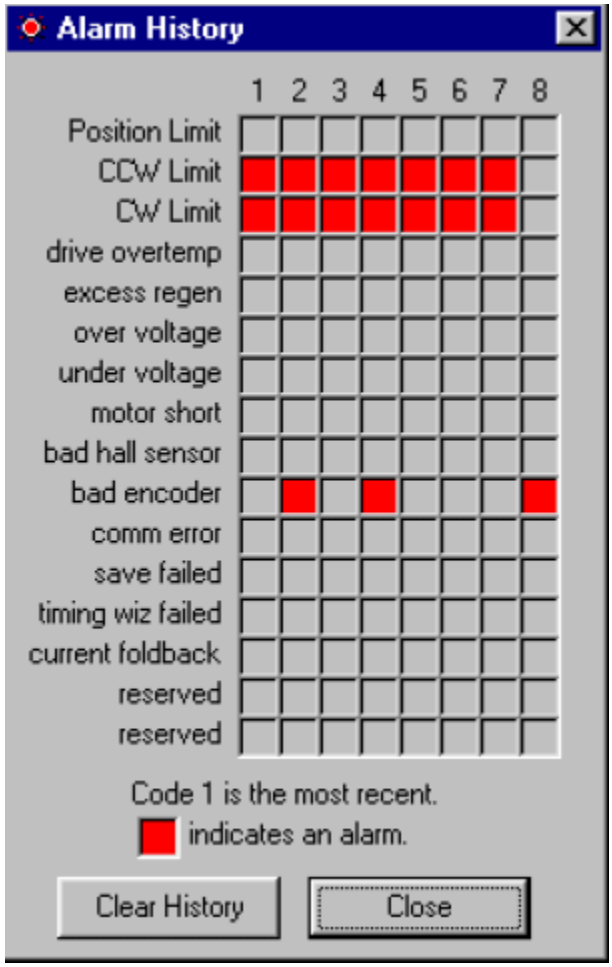

If something goes wrong with your servo system, you're going to want to know about it. If you are still working in "development mode" (i.e. the drive is still connected to your PC with the Mis Programmer software running) when a fault occurs, you'll get an on screen message. If you have a BLU servo drive with the latest firmware, you'll see a dialog like the one to the right detailing any and all errors and providing you with an option to clear the fault.

You'll certainly want your Mis program to take action if a servo fault occurs. The default action is to halt the motor and freeze your program. The drive will then display an alarm code on the front panel. Front panel alarm codes are shown on the next page.

You may also choose to branch to a program line if a servo fault happens. That way you alert the machine operator via the option MMI or you can automatically send a hardware signal using the Fault output (Y3 on the main board). If you are able to fix the problem, the servo can be re-enabled by the Change Tuning instruction if the Servo On option is selected.

If your drive has been operating "stand alone" and errors have occurred, you can connect the drive to the Mis Programmer and have a look at the eight errors. Just power up the drive with Mis Programmer present and click on Alarm History.



## *Front Panel Error Codes*

**BL series drives** show errors by flashing the power LED.

Position errors are shown as an “SOS’ pattern. (For the nautically-challenged, SOS is an endlessly repeating pattern of three long flashes followed by three short flashes.)

An unexpected encounter with an end of travel limit results in the power LED flashing at a constant rate.

Subroutine stack errors also use the constant rate error pattern.

BLU-DC servo drives use red and green LEDs to indicate alarm codes. In the event of an error, the green LED on the main board will flash one or two times, followed by a series of red flashes. The pattern repeats until the alarm is cleared.

| Code | Error |
|---|---|
| 1 red, 1 green | position error exceeds fault limit |
| 2 red, 1 green | ccw limit |
| 2 red, 2 green | cw limit |
| 3 red, 1 green | drive internal temperature exceeds 85°C |
| 3 red, 2 green | motor over temperature |
| 4 red, 1 green | power supply voltage is more than 55 VDC |
| 4 red, 2 green | power supply voltage is less than 18 VDC |
| 5 red, 1 green | over current / short circuit |
| 5 red, 2 green | current foldback - peak current $I_2T$ time exceeded |
| 6 red, 1 green | bad commutation(Hall) signal |
| 6 red, 2 green | bad encoder signal |
| 7 red, 1 green | serial communication error |

**BLU-AC servo drives** have a seven segment LED that displays alarm codes using numbers and letters, as shown here. Fault codes blink and disable the motor. Alarms do not blink and do not disable the motor.

| Code | Meaning |
|---|---|
| P | Fault: servo position error limit exceeded |
| L | Alarm: the ccw limit has been triggered |
| J | Alarm: the cw limit has been triggered |
| t | Fault: the Drive PCB temperature has exceeded 75° C. This will cause the drive to fault and cannot be cleared until the temperature drops below the limit. |
| O | Overvoltage fault: the DC Bus voltage exceeded 400 Volts |
| U | Under voltage alarm: the DC Bus voltage has gone below 100 Volts. |
| C | Overcurrent/short circuit fault: the motor phase current has exceeded the 20 amps rms |
| c | Peak current alarm: peak current I2T time exceeded, current temporarily reduced to continuous setting |
| H | Commutation fault: hall sensor inputs are incorrect. May indicate a failed sensor or bad cable. |
| E | Encoder fault: encoder not providing correct signals or single ended encoder used but not selected. |
| F | Flash or eeprom memory fault |
| r | Regeneration fault: while attempting to "dump" power into the regeneration shunt resistor the device limits were reached, or there is no regeneration shunt resistor attached. |
| d | servo disabled |
| o | Mis subroutine stack overflow |
| u | Mis subroutine stack underf low |

- Serial communication overrun or bit rate error
h Attempt to use drive with outdated version of Mis Programmer
1 The drive is in positioning mode. This is the mode used when performing "point to point" moves.
2 The drive is in velocity or jog mode.
3 The drive is in torque control mode.
4 The drive is in Step and Direction Control Mode. This is the mode used for "Encoder Following" or Positioning using Step and Direction inputs.
5 Mis positioning mode. You will see this indication most of time your Mis program is operating.

**STAC6-Mis** has a two color (red/green) LED to indicate status and alarm codes. A flashing green LED indicates that the drive is operating normally with the motor energized. Solid green means the motor is disabled.

In the event of an error, the green LED on the main board will flash one, two or three times, followed by a series of red flashes. The pattern repeats until the alarm is cleared.

| **Code** | **Error Description** |
|---|---|
| 1 red, 1 green | motor stall(when using option encoder) |
| 1 red, 3 green | subroutine stack overflow |
| 2 red, 1 green | ccw limit |
| 2 red, 2 green | cw 超限 |
| 2 red, 3 green | subroutine stall overflow |
| 3 red, 1 green | drive internal temperature exceeds 85℃ |
| 3 red, 2 green | excess regen |
| 4 red, 1 green | power replay overvoltage |
| 4 red, 2 green | power replay undervoltage |
| 4 red, 3 green | bad instruction in Mis Program(memory or software error |
| 5 red, 1 green | over current/short curcirt |
| 5 red, 2 green | motor resistance out of limit |
| 5 red, 3 green | Mis firmware is incompatible with DSP firmware |
| 6 red, 1 green | open motor widing |
| 6 red, 2 green | error encoder signal |
| 7 red, 1 green | serial communication error |

**Other Mis drives** with firmware 2.20 or later:

| **Code** | **Error Description** |
|---|---|
| 1 red, 0 green | limit |
| 1 red, 3 green | subroutine stack overflow |
| 2 red, 3 green | subroutine stack underflow |
| 4 red, 3 green | bad instruction in Mis program(memory or software error |
| 7 red, 1 green | serial communication error |

## *Using the Optional MMI*

Mis drives are available with an optional MMI (Man Machine Interface), sometimes called an operator panel. The MMI attaches to the same RS-232 port that you use to connect to your PC, using the same cable. The MMI has a four line liquid crystal display (LCD) and 20 keys for entering data. There are seven things you can do with the MMI:

1) You can display a message on the LCD. You might want to identify your machine ("ABC Bottle Filling Co. Model 20") or display a status message ("Machine Running - Status OK").

2) You can pause your program until the user presses ENTER. For example, if you were applying preprinted labels, eventually you'll want to halt the process until the operator loads a new roll of labels.

3) The MMI can ask the user to make a decision. For example, you might want to offer the user an option, like changing set up parameters, that can be responded to by pressing the yes or no keys.

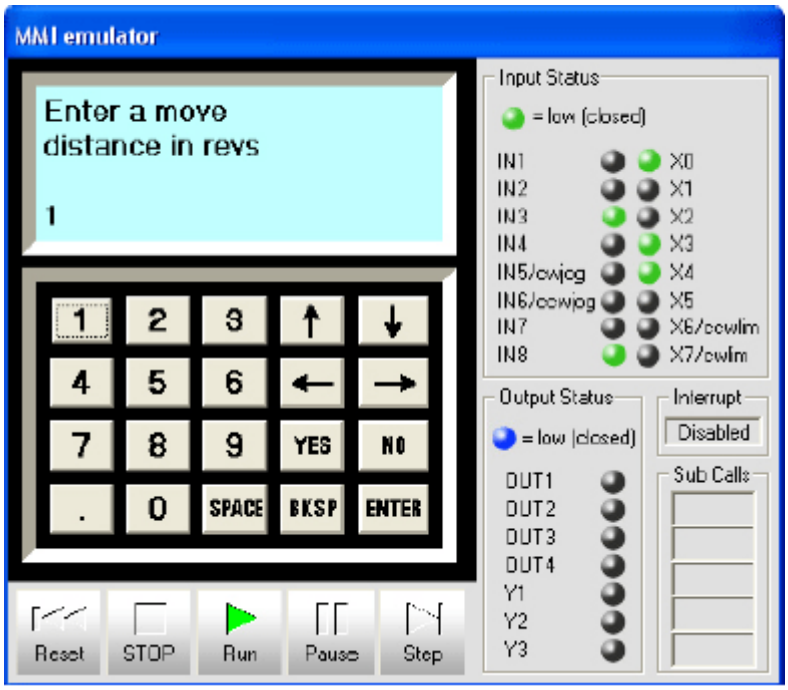
4) The user can be asked to enter a move distance. If you want to build a machine that feeds out material and then cuts it off, the operator can specify how long the resulting material will be.

5) The user can be asked for a move speed. This option allows the operator to adjust a feed rate, flow rate or other motor speed related setting.

6) The user can be asked for a repeat count. You can let the user set the number of parts that are processed. You can also combine a repeat loop with a Wait Time instruction to adjust dwell time.

7) You can display a menu, wait for the user to press a numeral key, then branch to a corresponding program line. Any or all of the keys 1 - 8 can be used, each with it's own branch address.

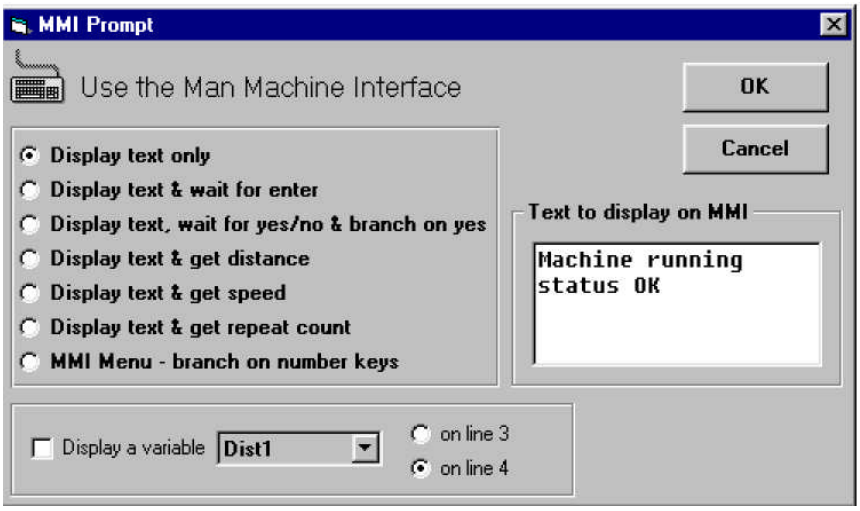
If you are thinking ahead at this point, you might ask "If the MMI plugs into the same port as the PC, how can I run a program from the PC that uses the MMI?" Like most features of the Mis software, it's simple. If you press the Execute button and the program in your drive contains any MMI instructions, you will see a different execute box on your screen. The MMI execute box looks and acts like the real MMI: it will display messages, and you can click on the buttons to enter data. Like the other execute box, there is a display showing the status of inputs and outputs, and a control panel that allows you to interrupt, single step, or restart your program at any time.




We provided the emulated MMI to save you the expense of a second RS-232 port on your Mis Indexer. It also allows you to try out the MMI before buying one.

***Note: If your indexer/drive firmware is prior to 1.40, you won't see the status display, because your drive is not able to provide real time status information to the PC. Instead, you'l just see the MMI Emulator.***

## How to display a message on the MMI

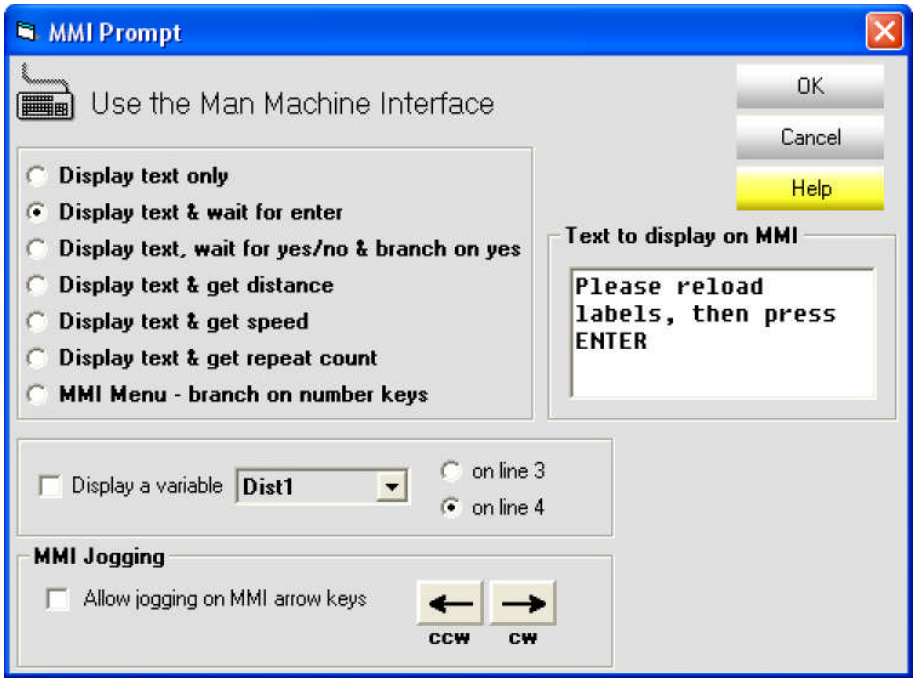


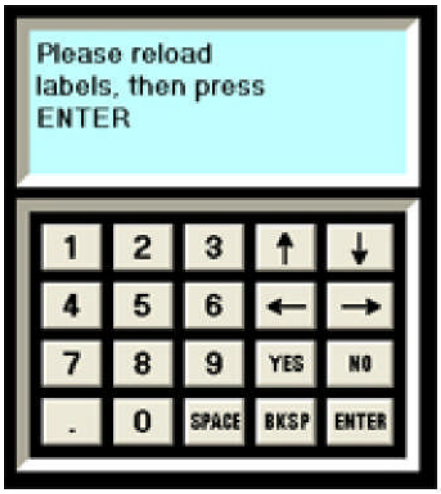
- Click on a program line icon
- Select the MMI Prompt instruction
- Type “Machine Running Status OK” in the text box
- Select the “Display Text Only” option button
- Click the OK button
- The message will stay on the LCD until another instruction uses the MMI.

If you have drive firmware 2.10 or later, you can also display an MMI variable on line 3 or on line 4. Just select the variable - the Mis Programmer software will determine what type of data it is (speed, distance or count) and apply the proper scaling.

You can also display a variable and the current repeat loop count. For example, you may want to display the total counts programmed and the present count. To do this, check the “Display a Variable” box, select the variable Count1 and choose line 3. Later in your program, when you set up the Repeat instruction, choose the same variable (Count1) and select “display loop count on MMI line 4”.

## *How to pause until user presses ENTER*



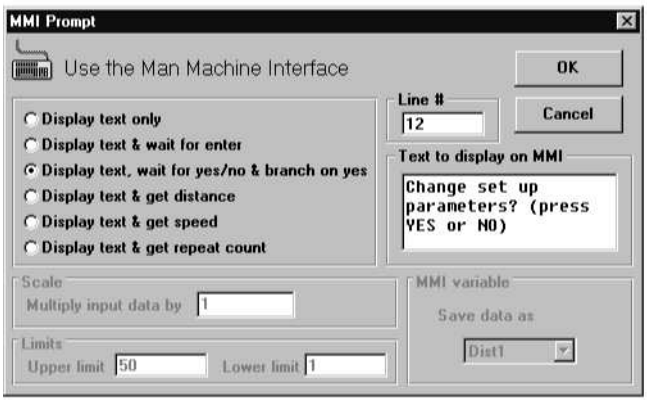

- Select the MMI Prompt instruction
- Type “Please reload labels, then press ENTER” in the text box
- Select the “Display text & wait for enter” option button
- If you want the user to be able to jog the motor using the MMI arrow keys, check the box marked “Allow jogging on MMI arrow keys.”
- Click OK

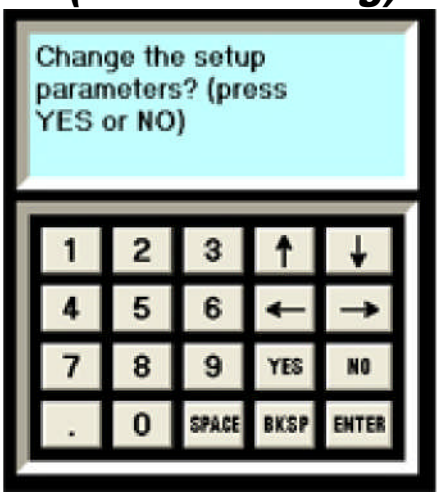
**Multiple Jog Speeds**
Drive firmware version 2.08 and later allows you to choose from multiple jog speeds. You can still use the “global” jog speed, as specified on the main screen. You can also select a fixed “local” jog speed that can be unique to each Wait Input instruction. This allows you to build a program with fast (rapid traverse) and slow (creep) jog speeds.

You can also select an MMI variable as the speed. This, in combination with an MMI Speed instruction, allows the operator of your system to enter one or more jog speeds.

**Display Variable**
If you have drive firmware 2.10 or later, you can also display an MMI variable on line 3 or on line 4. Just select the variable - the Mis Programmer software will determine what type of data it is (speed, distance or count) and apply the proper scaling.

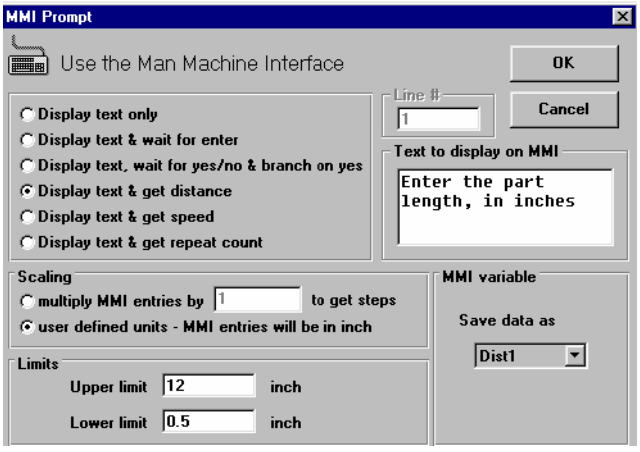
## How to let the user make a decision (MMI branching)

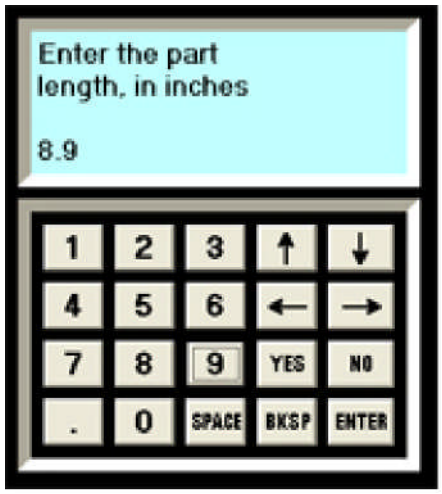



- Put an MMI Prompt instruction on line 1
- Type “Change setup parameters? (press yes or no)” in the text box
- Select the option button “Display text, wait for yes/no & branch on yes”
- In the line # box, type 12
- Click OK
- Starting on program line 12, place your parameter setting instructions.
- At the end of your parameter setting instructions, place a Go To line 2 instruction.

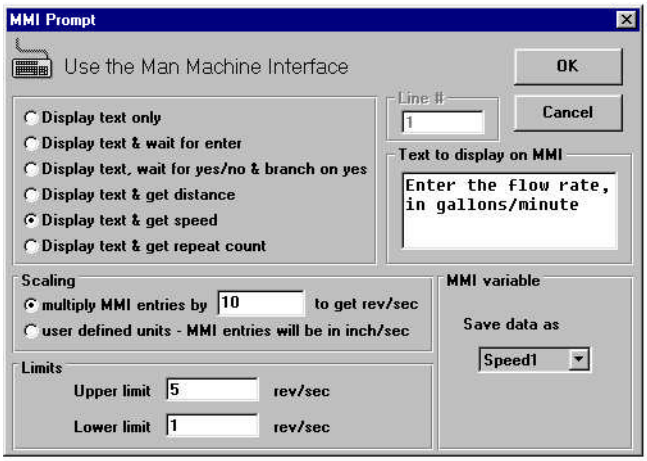
## How to ask the user for a move distance

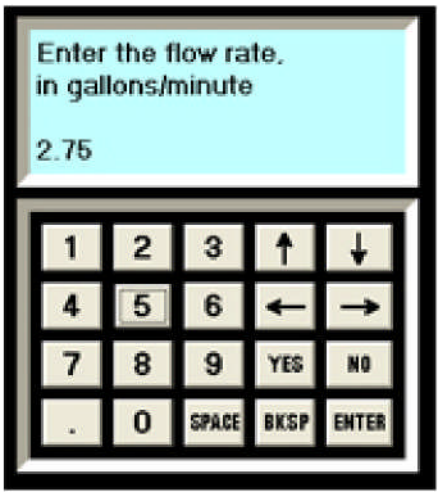



- Select an MMI Prompt instruction
- Type “Enter part length, in inches” in the text box
- Select the option button “Display text and get distance”
- Enter a scale factor, or, if you’re using User Defined Units, select “user defined units - MMI entries will be in inch” for automatic scaling.
- Enter upper and lower limits (in this example, we want to allow the operator to enter distances between 0.5 and 12 inches)
- Select an MMI variable to store the distance in (we chose Dist1 this time, but any of the eight MMI variables is acceptable for storing any type of data)
- Later in your program, you’ll need a Feed instruction (Feed to Length, Feed & Return, etc) that uses the Dist1 variable for distance.

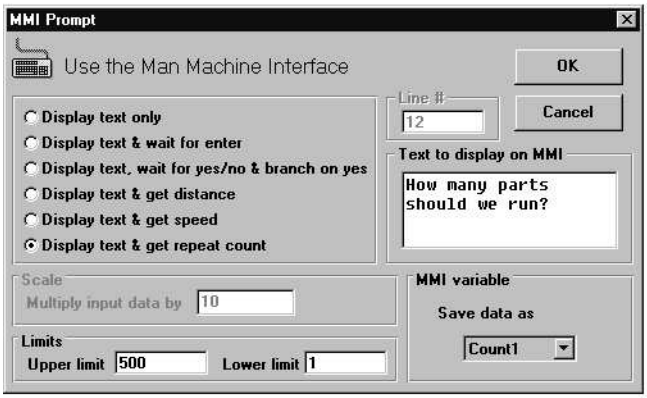
## How to get a speed from the user


MMI Prompt
Use the Man Machine Interface
OK
Cancel
Line #
1
Display text only
Display text & wait for enter
Display text, wait for yes/no & branch on yes
Display text & get distance
Display text & get speed
Display text & get repeat count
Text to display on MMI
Enter the flow rate, in gallons/minute
Scaling
multiply MMI entries by 10 to get rev/sec
user defined units - MMI entries will be in inch/sec
MMI variable
Save data as
Speed1
Limits
Upper limit 5 rev/sec
Lower limit 1 rev/sec




- Select an MMI Prompt instruction
- Type “Enter the flow rate, in gallons/minute” in the text box
- Select the option button “Display text and get speed”
- Enter a scale factor (in this example, one gallon/minute equals 10 revolutions/sec of the motor)
- Enter upper and lower limits (in this example, we want to allow the operator to enter flow rates between 1 and 5 gal/min)
- Select an MMI variable to store the speed in (we chose Speed1)
- Later in your program, you’ll need a Feed instruction (Feed to Length, Feed & Return, etc) that uses the Speed 1 variable for speed.

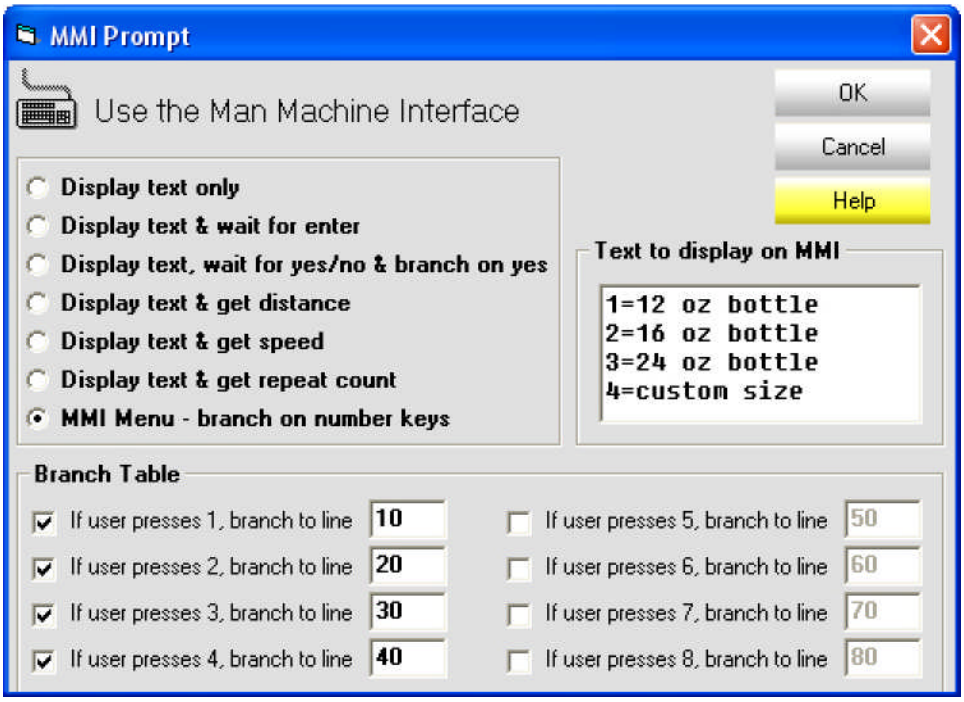
## How to get a repeat count from the user



- Select an MMI Prompt instruction
- Type “How many parts should we run?” in the text box
- Select the option button “Display text and get repeat count”
- Enter upper and lower limits (in this example, we want to allow the operator to enter a number between 1 and 500)
- Select an MMI variable to store the count in (we chose Count1)
- Later in your program, you’ll need a Repeat instruction that uses the Count1 variable for the repeat count.

## How to create an MMI Menu

Note: You need firmware version 1.41 or later to execute the MMI Menu function.


MMI Prompt
Use the Man Machine Interface
OK
Cancel
Help
Display text only
Display text & wait for enter
Display text, wait for yes/no & branch on yes
Display text & get distance
Display text & get speed
Display text & get repeat count
MMI Menu - branch on number keys
Text to display on MMI
1=12 oz bottle
2=16 oz bottle
3=24 oz bottle
4=custom size
Branch Table
If user presses 1, branch to line 10
If user presses 2, branch to line 20
If user presses 3, branch to line 30
If user presses 4, branch to line 40
If user presses 5, branch to line 50
If user presses 6, branch to line 60
If user presses 7, branch to line 70
If user presses 8, branch to line 80

- Select an MMI Prompt instruction
- Select the option button “MMI Menu...”
- Type your menu text in the text box (you can enter up to four lines)
- Check the boxes indicating which numeral keys you want to use (in this example, we used 1,2,3 and 4)
- Assign a program line number to each key (we used 10, 20, 30 and 40)
- Later in your program, you’ll need to put instructions at each of the lines you’ve specified. These are the instructions that will execute if the user presses the appropriate key on the MMI. For example, when the user presses ‘1’, the program will branch to line 10.

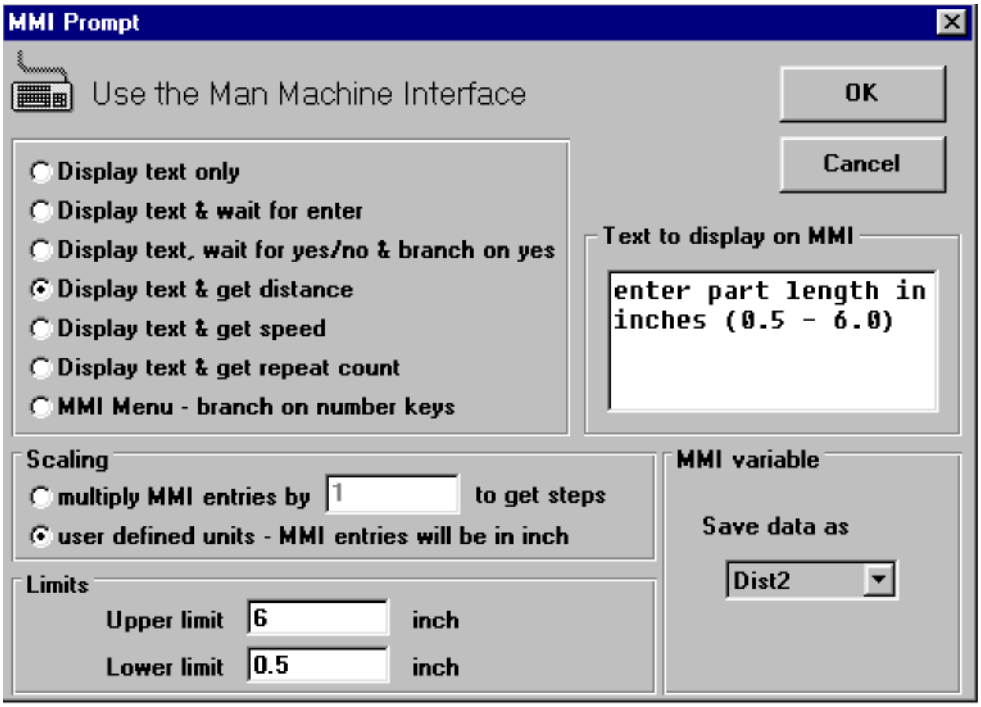
Note: if you add extra spaces to your display text to get the look “just right”, watch out: sometimes the Mis

## Making Your Move

### MMI Prompt

The MMI Prompt instruction is used with the optional MMI (Man Machine Interface). MMI prompts allow your program to display messages on the MMI screen, and can gather data from the operator to be used by other instructions. The MMI can also pause the program until the user presses the ENTER button. It can allow the user to make a decision, then press the YES or NO button. If the user presses YES, the program branches to another program line. If the user presses NO, the program goes to the next line.

If you just want to display a message, such as “Machine Running - Status OK”, put an MMI Prompt instruction in your program at the point where you want the message to appear. Check the option button marked “Display Text Only” and type in your message. Once the MMI Prompt instruction has been executed, the message will stay on the screen until changed by another instruction that uses the MMI display.


MMI Prompt
Use the Man Machine Interface
OK
Cancel
Display text only
Display text & wait for enter
Display text, wait for yes/no & branch on yes
Display text & get distance
Display text & get speed
Display text & get repeat count
MMI Menu - branch on number keys
Text to display on MMI
enter part length in inches (0.5 - 6.0)
Scaling
multiply MMI entries by 1 to get steps
user defined units - MMI entries will be in inch
MMI variable
Save data as
Dist2
Limits
Upper limit 6 inch
Lower limit 0.5 inch

If you want the operator of the machine you’re building to be able to change parameters like distance, speed or repeat count, you’ll need an MMI Prompt to ask the user for data and to store it in nonvolatile memory. In this case, click on the option button for the type of data you want: distance, speed or repeat count.

You’ll need to set upper and lower limits. The MMI Prompt instruction will check the data that’s entered against the limits you’ve specified, and tell the user if a value is out of range. For ex-ample, if you set the MMI Prompt to gather a repeat count, and you’ve set the upper and lower limits to 100 and 1, the instruction
will not accept any value bigger than 100 or smaller than 1.

You also must tell the MMI Prompt instruction where to store the data in nonvolatile memory. There are eight locations to choose from. They are named Dist1, Dist2, Dist3, Speed1, Speed2, Count1, Count2 and Count3. Remember where you told the MMI Prompt to put the data. When you set up an instruction to use data from an MMI variable, you must tell that instruction which variable to use (Dist1, Dist2, etc.)

Programmer removes those spaces when you reopen the instruction dialog. It is safer to format your text using other characters, like '.' (period) or '_' (underscore).

For example, If you want the operator to be able to set the number of parts your machine produces in a given run, put an MMI Prompt instruction in your program to ask for a repeat count and to save it as Count1. Some-where else in your program you'll set up a Repeat loop to process the parts. The loop will start with a Repeat instruction, one that you've configured to get its repeat count from the MMI variable Count1. You can even display the loop count on the MMI as your program runs. (Note: you need an indexer drive with firmware version 1.40 or later to display the loop count on the MMI.)

**Scaling**
The Mis indexers work internally in steps and revolutions per second. The MMI Prompt can accept data from the user in other units (like inches or inches/sec) and automatically scale the data to internal units. There are two ways to do that.

The easiest method of scaling is to set up user units on the main screen. That way your entire program can be entered in your units. See page 14 for an explanation of user defined units.

Another way to scale user entries is to enter a scale factor directly into the MMI Prompt dialog box. That way you can use different scale factors in different MMI Prompts.

Scaling is only available when gathering distance or speed data.

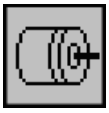
**Other Uses of MMI Prompts**
If you want to pause your program until the user presses the ENTER key on the MMI, choose the option marked "Display text & wait for enter." If you wish, you can allow the operator to use the MMI arrow keys for jogging. (Note: you need an indexer drive with firmware version 1.40 or later to do MMI arrow key jogging.)

To allow the user to make a decision, select "Display text, wait for yes/no & branch on yes." Be sure to enter a line number in the Line # box. The program will jump to that line if the user presses YES. If the user presses NO, the program will execute the next line after the MMI Prompt.

The MMI Menu option lets you assign line numbers to as many as eight numeral keys, and display text on all four lines of the MMI. When the operator presses one of the numeral keys, the program branches to the corresponding line. This is an easy way to set up a menu driven system.

If you have drive firmware 2.08 or later, you can also display an MMI variable on line 3 or on line 4 if you are using the MMI Text Only or MMI Wait Enter options. Just select the variable - the Mis Programmer software will determine what type of data it is (speed, distance or count) and apply the proper scaling.

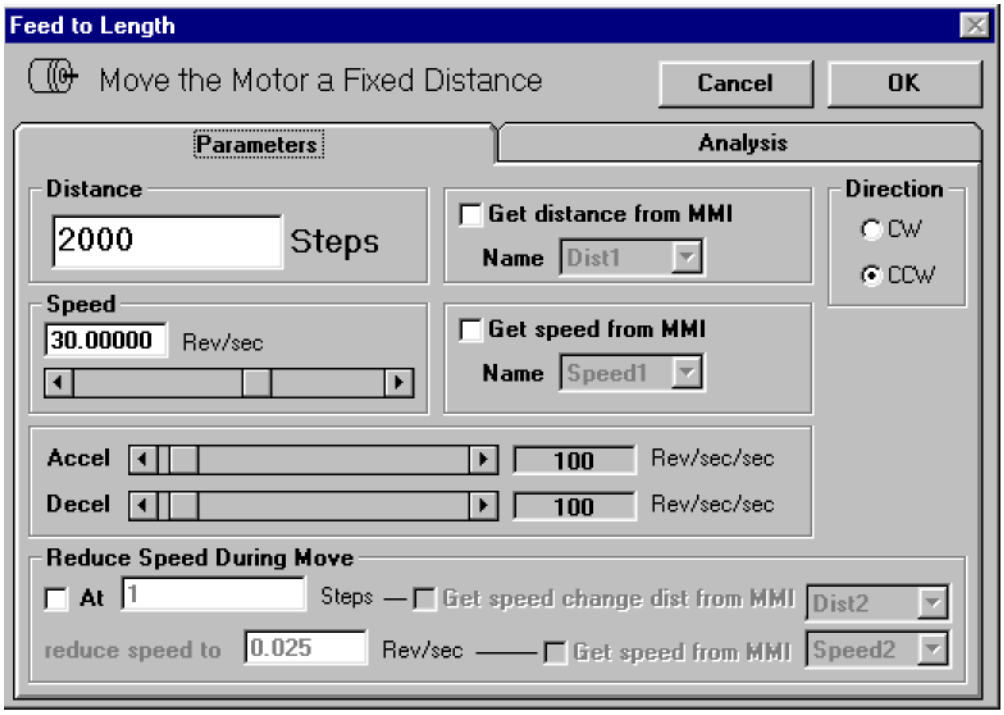
**Securing Your Machine Setup**
MMI Prompts are often used for machine set up. In some cases, it is necessary to prevent the machine opera-tor from changing these settings. The Mis Programmer does not provide password protection, but a mechanical "key switch" can be used for this purpose. The key switch should be wired to one of the programmable inputs of your Mis drive. (Please refer to your drive's hardware manual for wiring details.) Place an If Input instruc-



## *Feed to Length*

The Feed to Length instruction is used for point to point moves. If you just want to move the motor a fixed number of steps, this is the instruction to use. You can also use speed or distance data that was previously gathered by an MMI Prompt instruction.

When you click on the Feed to Length button in the Program Line... dialog box, you'll see the Feed to Length dialog box appear. This is where you enter the parameters for the move.




**Distance** - this is the number of motor steps you want to move. The maximum number is 16,000,000. If you select the check box marked "Get distance from MM I", you can choose one of the eight MMI variables as the distance. Please note that checking "Get distance from MM I" does not automatically make the Mis Indexer stop and ask the user for an entry. You'll need an MMI Prompt instruction somewhere else in your program for that.

**Speed** - this is the maximum speed you want the motor to go, in revolutions per second. You can set the speed anywhere between .025 and 50 rev/sec, in increments of .025 rev/sec. Servo drives have a speed range of .025 to 100 rev/sec. If you select the check box marked "Get speed from MMI", you can choose one of the eight MMI variables as the speed.

You can also reduce speed during the move by checking the box marked "Reduce speed during move." This is useful for applications where a tool may need to approach a work piece quickly, but slow down just before making contact.

tion in your program to jump over the MMI Prompts if the switch is in the locked position.

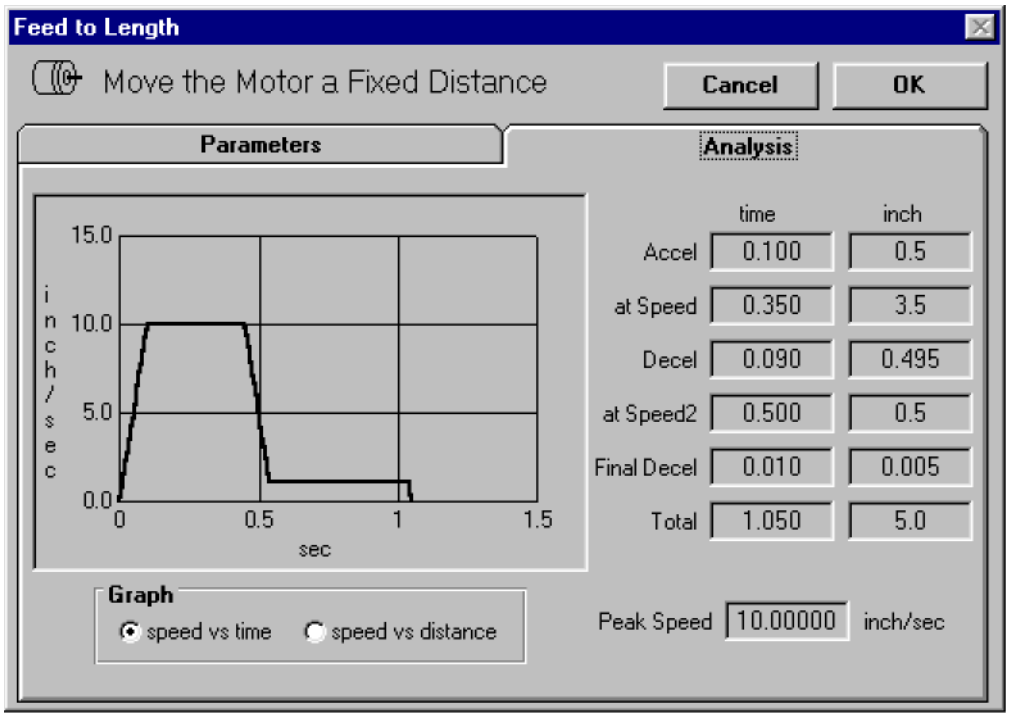
To see more examples of MMI Prompts, turn to page 23 or load the sample programs that are installed with the Mis Programmer.

**Accel** - motors cannot achieve a high speed instantly. The indexer-drive must gradually accelerate the motor to speed. The rate at which you can accelerate depends on the inertia of the motor and load, the torque available from the motor, and how fast you want it to go. You may need to experiment to find this out. The Mis Indexer has an acceleration range of 1 to 3000 revs/sec/sec.

**Decel** - this is the rate at which the drive decelerates to a stop at the end of the move. It's also the rate at which the motor reduces speed if you choose that option. The range is the same as for acceleration. Because friction encourages a motor to stop, you can usually set decel higher than accel.

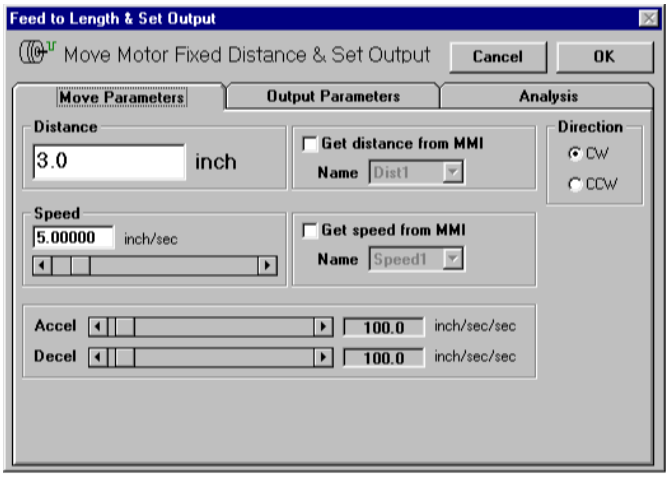
**Direction** - you can choose cw or ccw as the direction for the move. Just dot the appropriate circle by clicking on it.

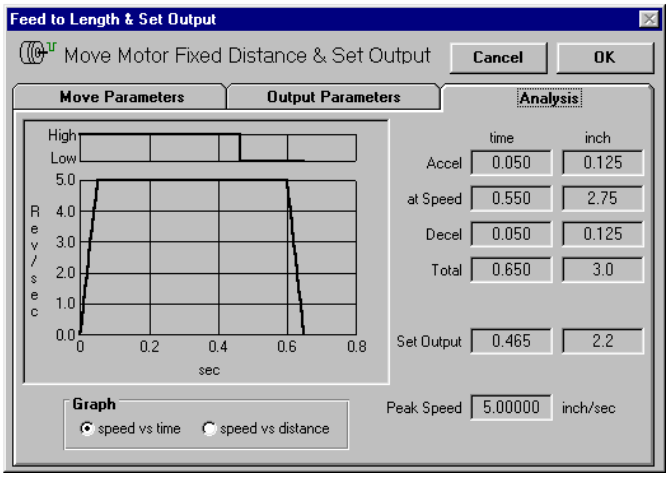
Analysis - Click on this tab to see a speed vs time graph of your move. It also provides some useful statistics about the move, such as the duration of the move and the portion of time spent accelerating and decelerating. You can also select a plot of speed vs distance.



## *Feed & Set Output*

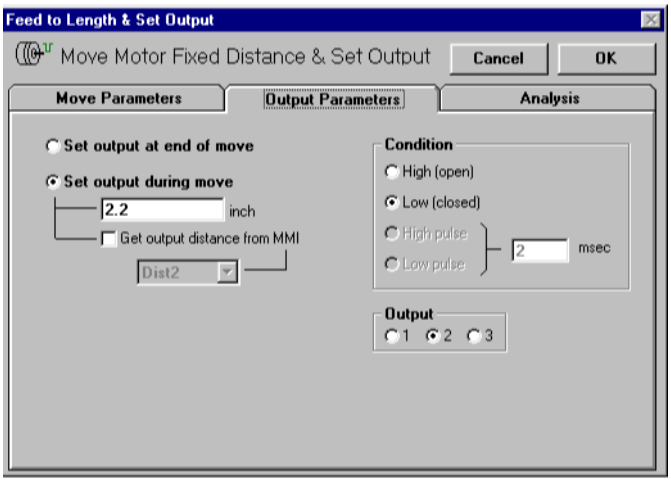
The Feed and Set Output instruction is provided for two reasons. First, it allows you to combine a Feed to Length instruction with a Set Output instruction, making your program shorter. Feed to Length and Set Output are frequently used in combination because you want your Mis product to signal another device when it finishes a move.

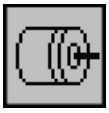



The second reason the Mis Programmer has a Feed & Set Output feature is for manufacturing throughput. The Mis indexer may be advancing a part which will then be processed by another device, for example feeding material to be cut off by blade. If the blade requires a little time to approach the material, you'd like to trigger the blade before you're done advancing the material. That way, you can process more pieces in the same amount of time.

Feed & Set Output allows you to set an output high or low at any point during the move.

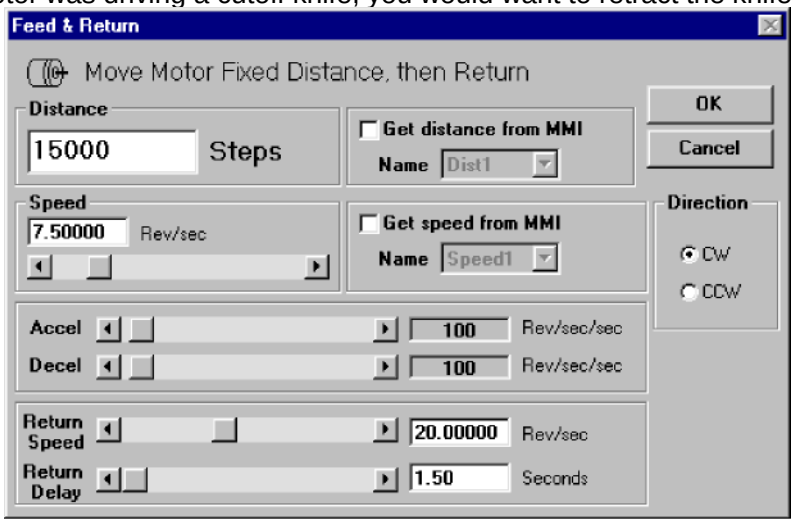




## *Feed & Return*

The Feed & Return instruction is used for point to point moves where you want to return to the starting point.

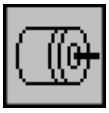
For example, if the motor was driving a cutoff knife, you would want to retract the knife after cutting.




***Feed & Return*** requires many of the same parameters as Feed to Length: distance, speed, accel, decel and direction. For explanations of these, please refer to the Feed to Length section of the manual.

You'll also need to set the return speed. The range is .025 to 50 revolutions per second. Servo drives have a top speed of 100 rev/sec. In the case of the cutoff knife, you might want to feed slowly, as the knife is cutting, then retract quickly. Thus, you would set the return speed higher than the forward speed.

***Return delay*** determines how long the Mis Indexer waits between the end of the feed move and the start of the return. This could, for example, give the machine time to remove a part before retracting. Since a motor and load need time to "settle out" after moving, you should not set return delay to less than 0.2 seconds unless you are certain that your motor and load settle more quickly than normal.

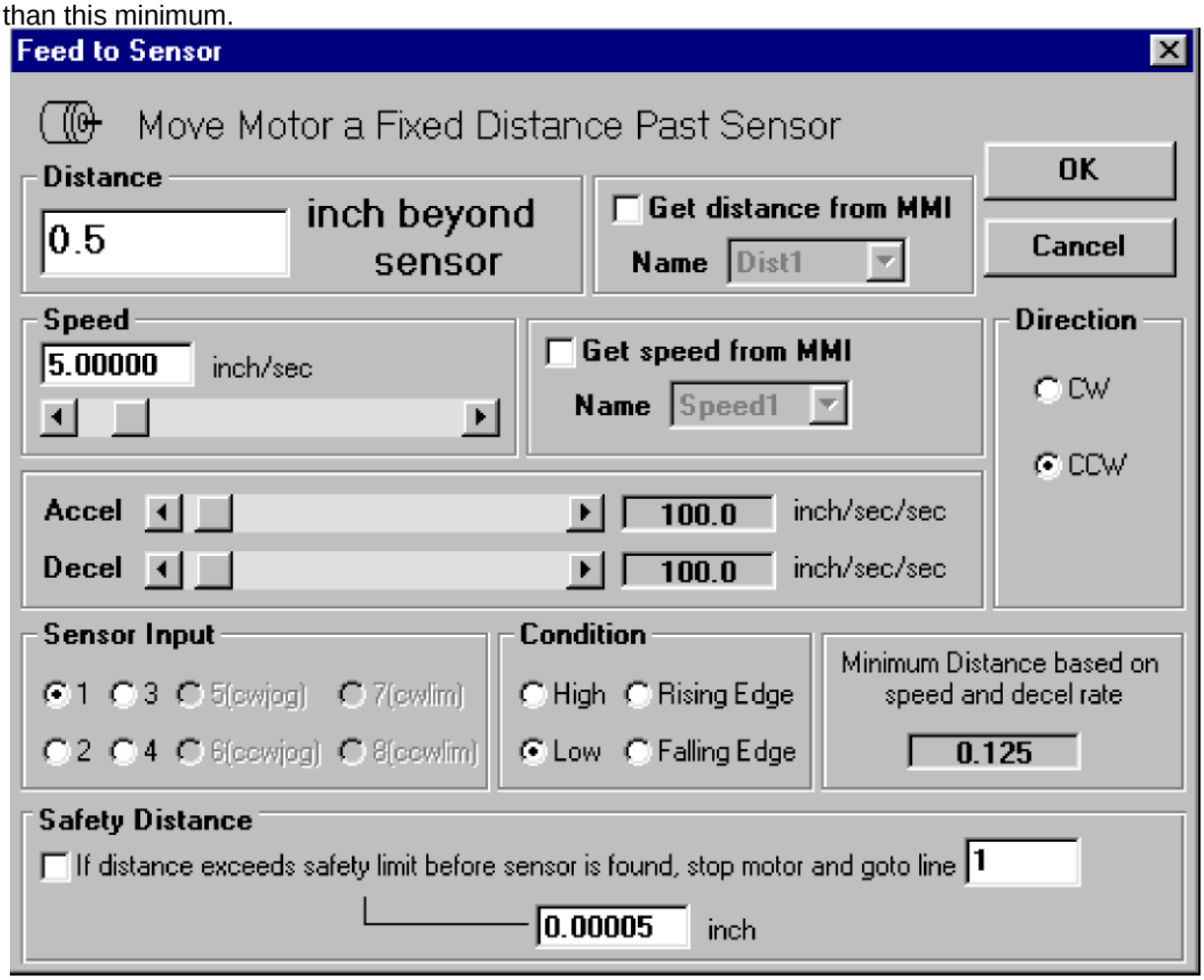


### *Feed to Sensor*

The Feed to Sensor instruction allows you to move the motor until an external event changes the state of an input.

One useful application for Feed to Sensor is when your motion distance varies. Let's say you are using a motor to dispense labels that come on a roll. You can't guarantee that the spacing of the labels is exact, so you don't want to simply feed out the same number of steps each time. Instead, you can put a sensor on the feed mechanism that "sees" the edge of each label and signals one of the Mis Indexer inputs to stop motion.

Feed to Sensor will ask for many of the same parameters as the other feed programs: Speed, accel, decel and direction. You also need to specify a distance. That's because the Mis Indexer must have enough space to decelerate to a stop once the sensor is tripped. The higher the speed, the longer it will take to stop. If the decel rate is increased, then the motor can stop in fewer steps. The "Minimum Distance" box tells you how many steps you must allow, based on the speed and decel rate that you've set. You can't set the distance to less than this minimum.



You'll also need to tell the Mis Indexer which input the sensor is wired to and what input condition to look for. The four input conditions are:

**High** - move until the specified input reaches a high signal state. This is the default state of an input if nothing is connected to it.

**Low** - move until specified input is at a low signal state.

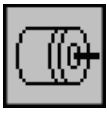
**Rising Edge** - move until the signal goes from low to high. This is similar to the high condition, but the difference is important. Let's say that you have a sensor wired to the Mis Indexer that will go high when you want motion to stop. However, the sensor signal stays high after motion is complete, going low sometime later. This often happens in labeling applications where there isn't much space on the roll between labels. If you choose high as your input condition, the Mis Indexer will complete the motion, then refuse to start again because the input signal is still high. If you choose rising edge, the Mis Indexer would proceed with the input voltage high and stop when the sensor signal goes from low to high again.

**Falling Edge** - the opposite of rising edge. Mis Indexer waits for an input voltage to go high, then low.

If you have a BLU-Mis drive, you will see more input choices, including X0, the encoder index. With Applied Motion motors, X0 reads high when you are on the index and low everywhere else.

If you are concerned about your load never reaching the sensor (for example, if your sensor may fail or the load might jam up), check the box marked "If distance exceeds safety limit...". You can then enter a safe distance and specify a line number to which the program will branch if it can't find the sensor. For example, if you are sensing labels on a roll, and they are supposed to be about 1 inch apart, enter a safety distance of 3 inches. Then enter 20 as the branch line. On line 20, you will put some kind of error recovery routine, like an MMI Prompt telling the operator to check the label stock.

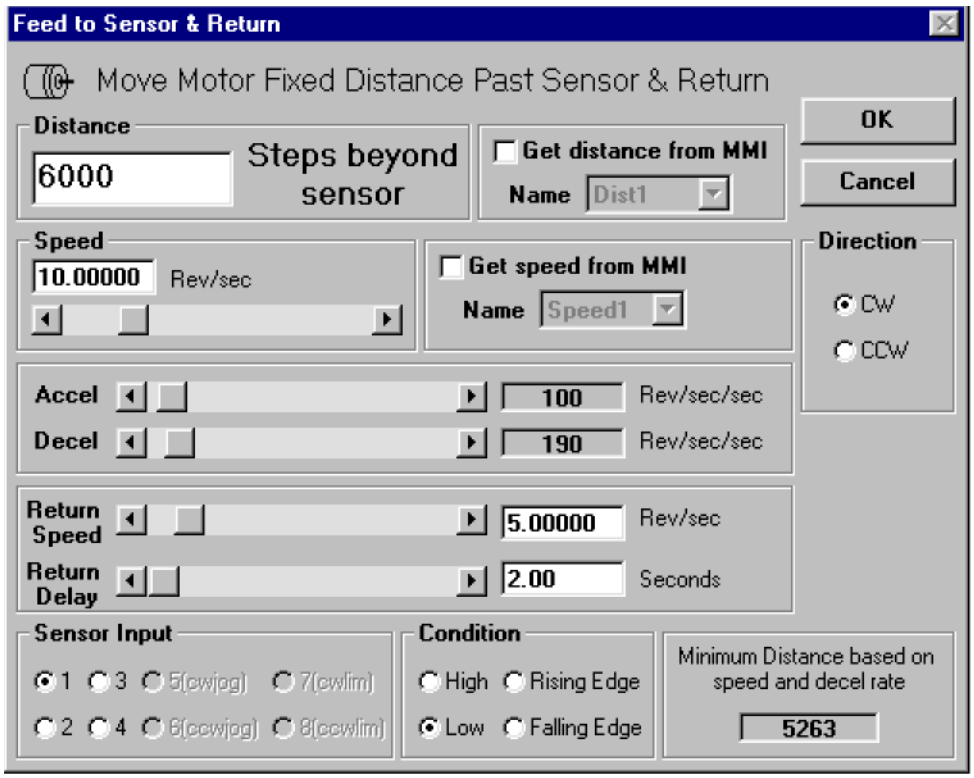
***Note: the Mis Programmer software wil not let you enter a move distance that is less than the minimum deceleration distance for your chosen speed and decel rate. However, if speed or distance is set by an MMI variable, no such error checking is performed. It is your responsibility to choose an upper limit of speed and lower limit of distance so that an operator cannot enter an incorrect value. Failure to do this may result in unexpectedly long moves.***



## *Feed to Sensor & Return*

The Feed to Sensor & Return instruction is just like Feed to Sensor, but after the move the motor returns to the starting point.

Most of the parameters are the same as Feed to Sensor, but two new ones are added: the return speed and the return delay.




One useful application of Feed to Sensor & Return is a variable distance application. If we were building a machine to cut fabric of different sizes, and the Mis Indexer was driving the cutoff knife, we might want to set a sensor at the end of the cut off stroke. That way, we can manually adjust for the width of material we happen to be using on a particular day without having to reprogram the Mis Indexer.

Each time it's triggered, the indexer-drive would feed until the knife trips the sensor, then return to the starting point.

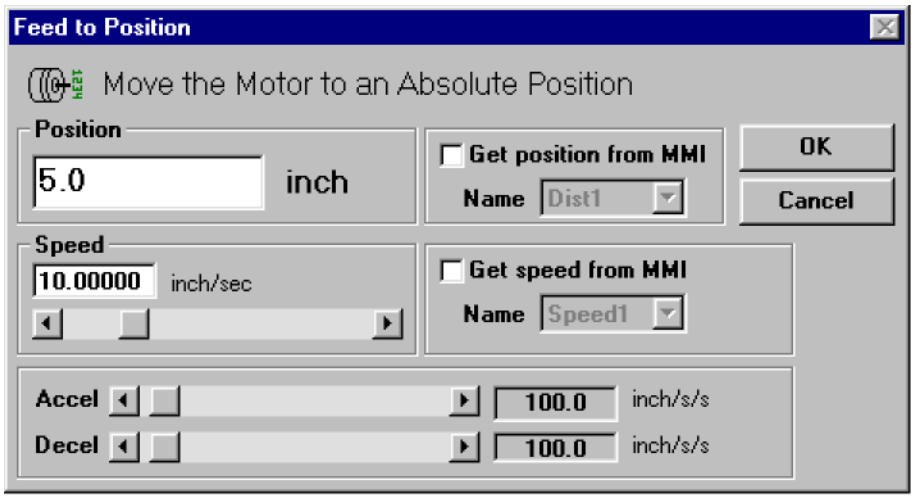
You should beware of one thing: the maximum distance for any program is 16,000,000 steps. That's the longest distance the Mis Indexer can track. If you move more than 16 million steps before you hit the sensor, the Mis Indexer will not return to the correct position. If this is a problem for you, consider selecting a lower microstep resolution. At 50,000 steps/rev, you would exceed the 16 million step limit after 320 revolutions. At 2000 steps/rev, you can go 8000 revs before exceeding the limit.

## *Feed to Position*

This instruction moves the motor and load from wherever they are to an absolute position. For example, if the load is at the 4 inch position and the program executes a Feed to Position 6 inches, the motor will move two inches clockwise. If the load was at the 10 inch position and you did a Feed to Position 6 inches, the move would be 4 inches counter clockwise.

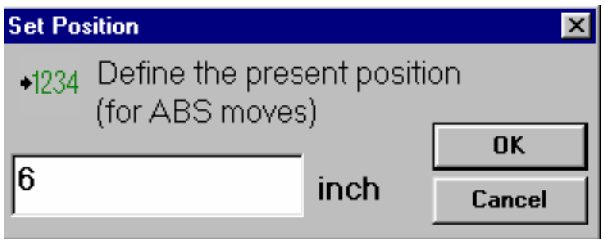
Feed to Position requires the usual move parameters: speed, accel & decel rates. Like other move instructions, speed can be recalled from an MMI variable, allowing it to be entered by the operator on the MMI panel.

Position can be a positive or negative number, and can be entered on the MMI. Please note that the MMI has no minus (-) key, so you can't enter a negative number on the MMI. You can avoid using negative absolute positions by using the Set Position instruction.



### *Set Abs Position*

This instruction allows you to define the present motor position as any absolute position you like. The Seek Home instruction automatically clears the absolute position counter when it's finished, defining the home position as 0. But you may want something else. Perhaps you want to think of the home sensor as being the 8 inch position, or 90 degrees, or whatever. Simply put a Set Position instruction after the Seek Home instruction, or anywhere else in your program where you want to define the absolute position.



## *Save Abs Position*

This instruction allows you to save the present absolute position to an MMI variable. This is useful if you want an operator to visually position your load, and then be able to return to that position later in your program.

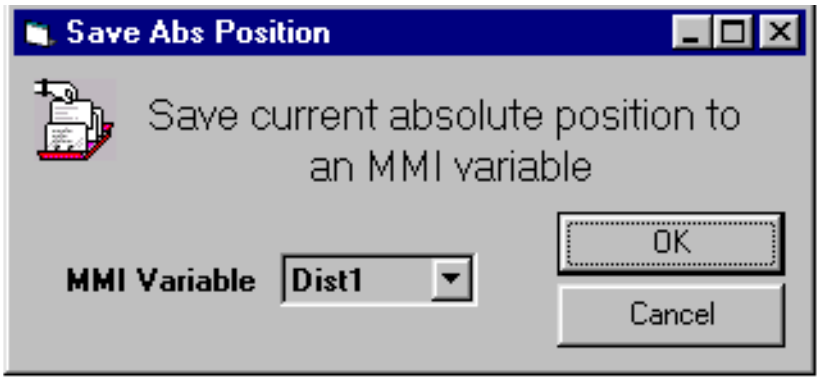
For a material handling application, you could create a program that uses the Wait Input or Hand Wheel commands to allow the operator to move your load into position. The operator would then press an ENTER button to exit the Wait Input or Hand Wheel instruction. If the next instruction is Save Abs Position, the load position that the operator carefully obtained is recorded in nonvolatile memory. Elsewhere in your program you can use a Feed to Position instruction to return the load there.

Since the Mis indexers support up to 8 MMI variables, you can save as many as eight different positions.

The sample program “LPdemo” demonstrates the “learning” of two positions.

Even though the positions that the indexer has “learned” will still be remembered the next day (because they are stored in nonvolatile memory), you will need to “home” the system each time it’s powered up. Otherwise, the absolute positions that you’ve saved don’t make any sense.

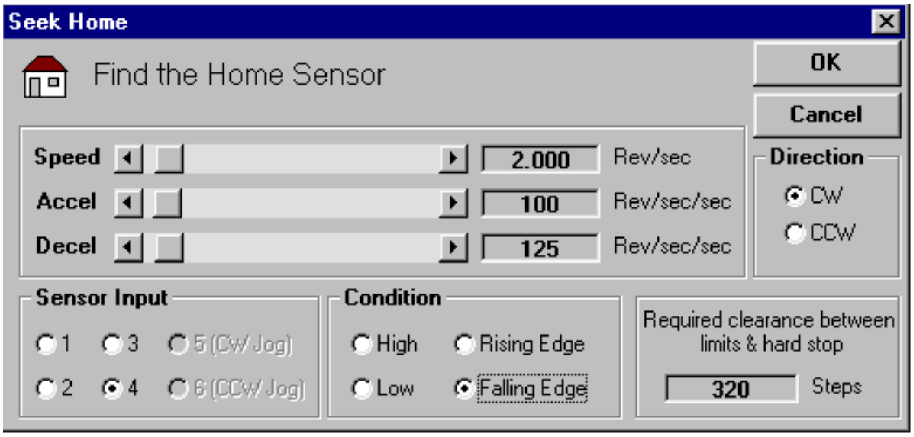
***Note: you need an indexer drive with firmware version 1.52 or later to use the Save Abs Position instruction.***



## *Seek Home*

The Seek Home instruction allows you to move the motor until a home sensor is found. The home sensor can be wired to any of the general purpose inputs.

Some applications require the motor to start from a certain position each time you turn on the power, but can't guarantee where it was left at the last power down. The solution is to wire a sensor to one of the Mis Indexer inputs and place a Seek Home command at or near the beginning of the program.




Seek Home will ask for many of the same parameters as the other feed programs: Speed, accel, decel and direction.

You'll also need to tell the Mis Indexer which input the sensor is wired to and what input condition to look for. The four input conditions are:

**High** - move until the specified input reaches a high voltage state. This is the default state of an input if nothing is connected to it.

**Low** - move until the specified input is at a low voltage state.

**Rising Edge** - move until the signal goes from low to high. This is similar to the high condition, but the difference is important. If you execute a Seek Home command to a high input and the load is already on the home sensor (causing the input to be high) then the load will not move. If you choose "rising edge" instead, the Mis Indexer will move the load to the edge of the home sensor.

If you need the load to be at the exact same position after each Seek Home command, choose Rising Edge or Faling Edge.

**Falling Edge** - the opposite of rising edge. Mis Indexer waits for an input voltage to go high, then low.

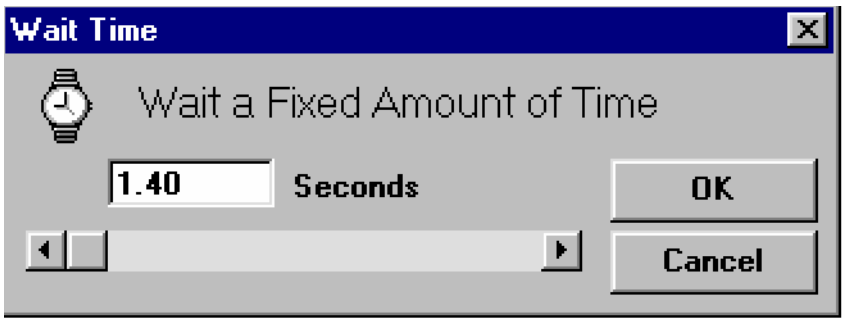
The Mis Indexer begins a Seek Home command by moving the motor in the direction you have specified. If the home sensor is found, the motor decelerates to a stop, then backs up to the sensor. If a limit is encountered before the home sensor is found, the Mis Indexer reverses the direction of motion and keeps looking for the home sensor.

You may have noticed a box in the lower right-hand corner of the Seek Home dialog box. This tells you how many steps the Mis Indexer needs to decelerate to a stop. The "Required Clearance" box tells you how much distance you must allow between the limit sensors and any hard stop, based on the speed and decel rate that you've set. If you don't allow enough clearance, the load may crash into something as it decelerates past a limit while seeking home. The higher the speed, the longer it will take to stop. If the decel rate is increased, then the motor can stop in fewer steps.

If you have a BLU-Mis drive, you will see more input choices, including X0, the encoder index. With Applied Motion motors, X0 reads high when you are on the index and low everywhere else.

### *Wait Time*

This is the simplest instruction. Just enter an amount of time, and the Mis Indexer will pause for that time

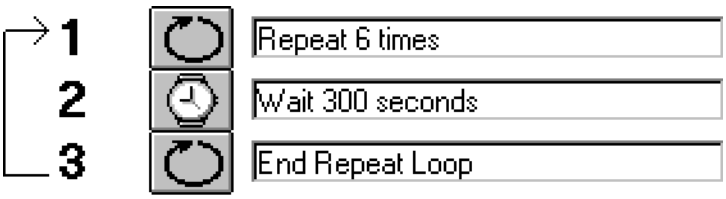



before proceeding to the next line in the program The range is 0.01 to 300 seconds.

What, you want to pause for more than 300 seconds? Did I hear you say 30 minutes? Okay, we can do that. That's the beauty of multiple line programs with a wide range of instructions - you're only limited by your creativity.

You can make the Wait Time instruction last longer by placing a repeat loop around it. Now I know we haven't talked about repeat loops yet, so we're going to skip ahead a little here. The first trick is to factor your 3 minute delay into two parts. 30 minutes is 1800 seconds, right? The most we can delay in one Wait Time instruction is 300 seconds. Okay, what if we delay for 300 seconds 6 times?

Your program would look like this:



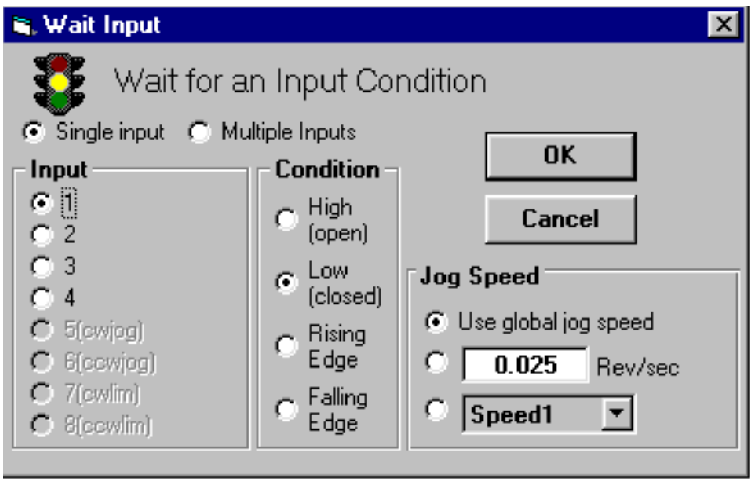

What's the limit? Well, a repeat loop can go 65535 times, so the maximum time you can delay is 65535 x 300 = 19.66 million seconds, or 5461 hours. Not long enough? Try two repeat loops, one inside the other. Now we're pausing for up to 40000 years. Wow!

## *Wait Input*

Rarely does a motion controller operate completely on its own, with no input from the outside world. In most cases, you'll need the Mis Indexer to wait for something to happen before it goes into motion. The Wait Input command is used for that. The Wait Input command is also the only instruction that allows you to jog the motor using the JOG CW and JOG CCW inputs.

The Wait Input instruction has two modes of operation: single input, where it only looks at one input, and multiple inputs where it can examine up to 8 inputs at once (16 inputs for BLU-Mis drives).

In Single Input mode, you must specify the input and the voltage condition to expect. The choices are:




**High** - Wait until the specified input is at a high voltage state. This is the state an input will be in if nothing is connected to it, so be careful if you use this condition. If a wire comes loose, you could end up with undesired motion.

**Low** - Wait until specified input is at a low voltage state. This happens when the input is conducting current. If you use a momentary contact switch (normally open type), this condition will occur when you press the button.

**Rising Edge** - Wait until the signal goes from low to high. This is similar to the high condition, but the difference is important. Let's say that your signal into the Mis Indexer is one that will go high when you want motion to occur. However, the signal remains high after the motion is complete, going low sometime later. If you choose high as your input condition, the Mis Indexer will complete the motion and start again because the input signal is still high when it finishes the first move. If you choose rising edge, the Mis Indexer will wait for the input voltage to go low, then high before moving.

**Falling Edge** - The opposite of rising edge. Mis Indexer waits for input voltage to go high, then low.
If you select "Multiple Inputs", the Wait Input instruction will scan multiple inputs at once to determine if the program should move on to the next instruction. The inputs can be configured as a binary sum (Wait for input 1 low or 3 high) or as a product (Wait for input 1 low and 3 high).
If you have a BLU-Mis drive, you will see more input choices, including X0, the encoder index. With Applied Motion motors, X0 reads high when you are on the index and low everywhere else.

**Multiple Jog Speeds**
Drive firmware version 2.08 and later allows you to choose from multiple jog speeds. You can still use the "global" jog speed, as specified on the main screen. You can also select a fixed "local"jog speed that can be unique to each Wait Input instruction. This allows you to build a program with fast (rapid traverse) and slow (creep) jog speeds.

You can also select an MMI variable as the speed. This, in combination with an MMI Speed instruction, allows the operator of your system to enter one or more jog speeds.

***Note: the condition LOW (or CLOSED) occurs when current is flowing through the input terminal. HIGH (OPEN) means that no current is flowing.***

## *Hand Wheel*

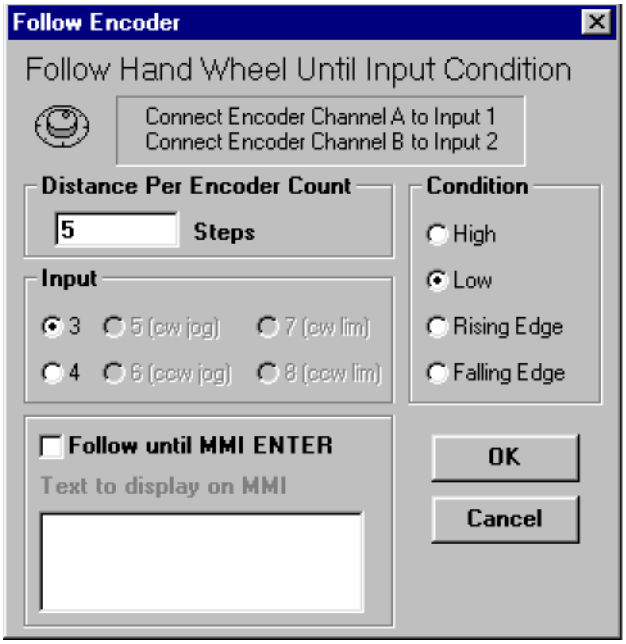
The Hand Wheel instruction is similar to Wait Input. You specify an input and a condition to be met, like “Input 4 low”, and the instruction continues executing until that happens.

The Wait Input instruction allows the user to move the motor using the CW JOG and CCW JOG inputs, or using the MMI arrow keys. Hand Wheel does the same thing, except that it positions the motor as you turn a CNC type hand wheel. The outputs of the hand wheel connect to inputs 1 and 2 of the Mis indexer. You can specify the “gearing” by telling the Mis how much motor distance to move each time the hand wheel “clicks” to the next position.

Hand wheels with 100 counts/rev work best. Call the factory for availability of a suitable hand wheel if you are interested in this feature.

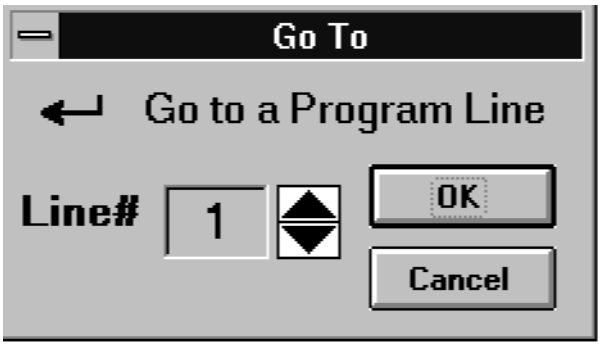
You could think of the hand wheel positioning as “digital jogging”: it allows the operator of a machine to achieve very precise adjustment of the load position.

***The Hand Wheel is not supported by the Mis-100.***



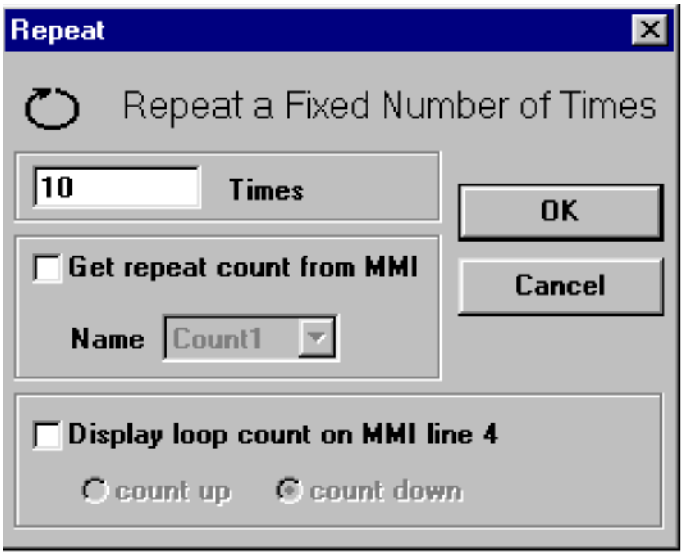
### *Go To*

The Go To instruction is used to make the indexer-drive jump to another line in the program. At the least, you'll need to have a Go To at the end of your program, to jump back to the beginning.




There is only one parameter to enter in a Go To instruction: the line number you want to jump to. Click on the spin button to increase or decrease the line number.

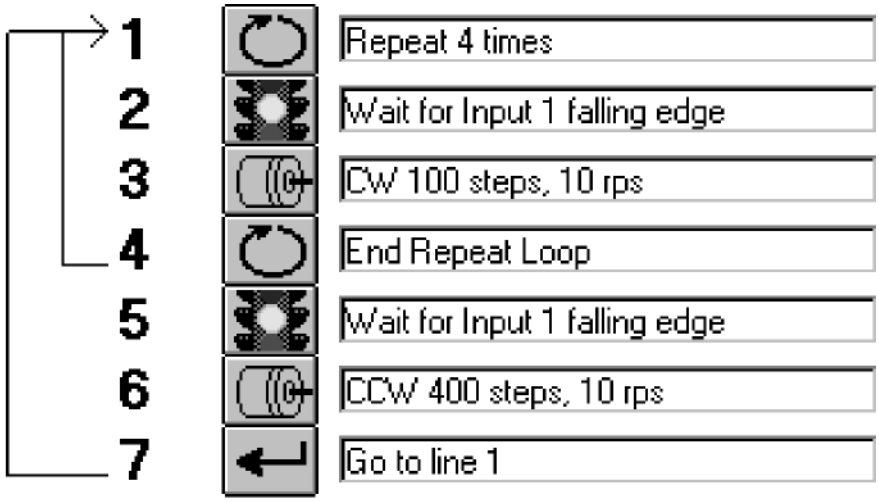
### *Repeat/End Repeat*




Sometimes you need to do the same thing several times, and you know in advance how many times that is. Re-peat loops allow you to repeat the instructions inside the loop up to 65,535 times.

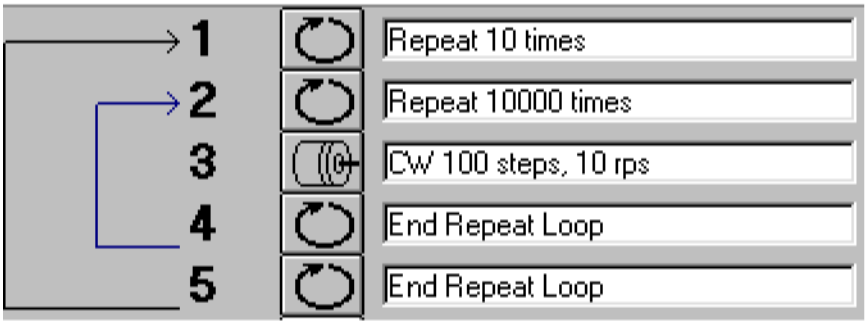
For example, let's say we are dispensing fluids into an array of containers. There are five rows and five columns of containers, each 100 steps away from the next. Each time we receive a trigger command, we want to move to the next position. After the fifth container is full, we must return to the first, 400 steps back.

The Mis Indexer that controls the X axis, or motion between columns, would be programmed as follows:
The program begins on line 1. There, we enter the repeat loop. The next four times, the Mis Indexer will wait for the voltage at Input 1 to fall, then move clock-wise 100 steps, taking the dispenser to the next container. After the fourth time, the Mis Indexer drops out of the loop into line 5. This time when Input 1 falls, the motor is moved 400 steps counter-clockwise, returning to the original position.




Sometimes you may need to repeat something more than 65,535 times. Let's say your task is to feed material into a cut off knife, and you want to run 100,000 pieces.

The best solution is to set up 2 loops,
one inside the other. The total number of cycles will be the number of repeats in the two loops multiplied together. 100,000 is 10,000 x 10, so we could set one loop for 10 and the other for 10,000.

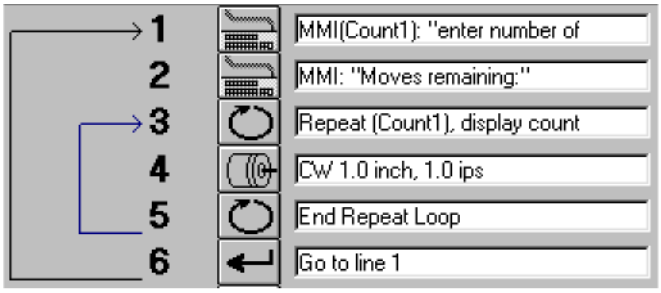


The *Repeat* instruction can also use data that was gathered and stored by an MMI Prompt instruction as the loop count. Just check the box marked “Get repeat count from MM I” and select a variable from the list.

For example, you could put an MMI Prompt in your program to ask for the number of parts to be processed and save that data as Count1. You would then set up the Repeat instruction to get the repeat count from the MMI variable Count1.

You can also display the loop count on the MMI as your program runs. You can count up (displaying the number of parts that have been processed, for example) or you can count down (showing the number of parts remaining.)

When the Repeat instruction displays the loop count on the MMI, it uses line 4. You may want to put an MMI Prompt to “display text only” just before the Repeat instruction telling the operator what the count means, as shown below.



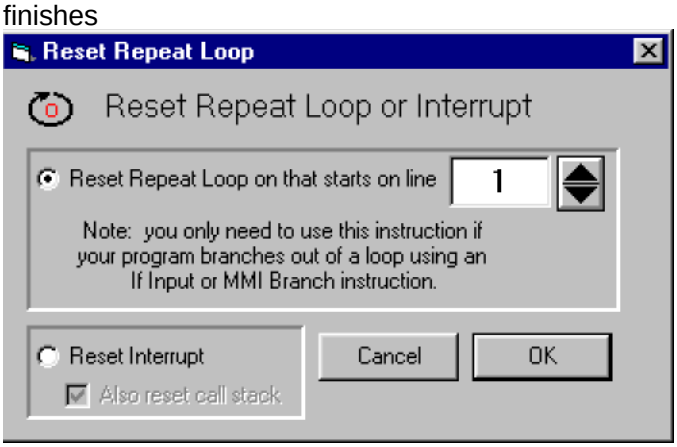

***Note: If you use an If Input instruction to exit a Repeat Loop, the loop does not automaticaly reset the next time you enter it. If you exit a loop by using an If Input instruction, you should use a Reset Repeat Loop instruction to reset the loop. Otherwise, the loop count resumes where it left off.***

## *Reset Loop or Interrupt*

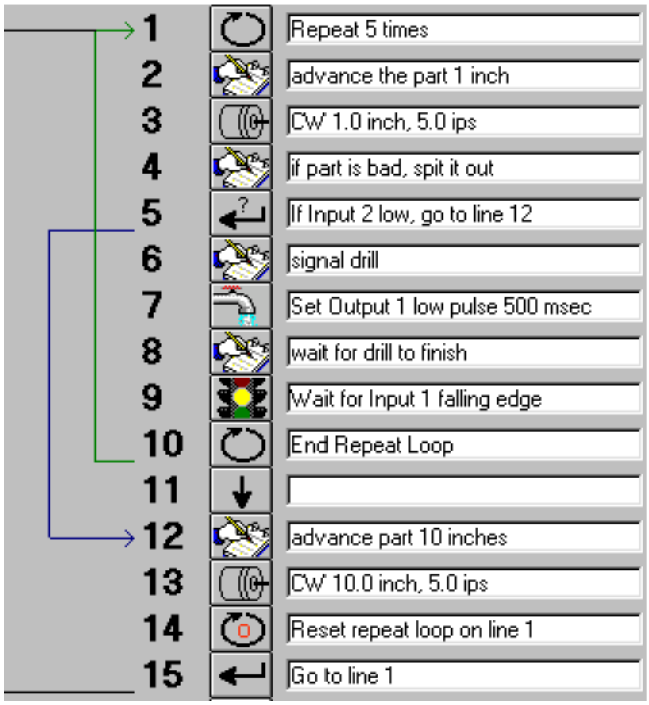
**Reset Loop Option**
Forces a Repeat Loop to reset its counter if it's been terminated by an If Input or Feed to Sensor with Safety Distance instruction.

Sometimes it is necessary to leave a repeat loop before it is completed. Say, for example, you have a repeat loop that is set up to fill 100 bottles with fluid. If the reservoir runs dry, you want to leave the loop. You can do this by putting an If Input instruction inside the loop, triggered by a fluid sensor. The If Input would branch outside the loop, perhaps to an MMI Prompt telling the machine operator to refill the reservoir.

Now suppose that 60 bottles have been filled, with 40 remaining. If you want the loop to "pick up where it left off", then simply branch back to the beginning of the loop (to the Repeat instruction) after the operator finishes


Reset Repeat Loop
Reset Repeat Loop or Interrupt
Reset Repeat Loop on that starts on line
1
Note: you only need to use this instruction if your program branches out of a loop using an If Input or MMI Branch instruction.
Reset Interrupt
Also reset call stack
Cancel
OK

refilling, and the loop will automatically fill the remaining 40 bottles.

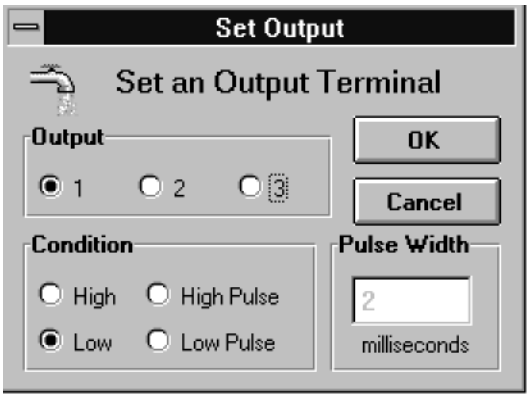

1 Repeat 5 times
2 advance the part 1 inch
3 CW 1.0 inch, 5.0 ips
4 if part is bad, spit it out
5 If Input 2 low, go to line 12
6 signal drill
7 Set Output 1 low pulse 500 msec
8 wait for drill to finish
9 Wait for Input 1 falling edge
10 End Repeat Loop
11
12 advance part 10 inches
13 CW 10.0 inch, 5.0 ips
14 Reset repeat loop on line 1
15 Go to line 1

On the other hand, what if you are drilling holes in parts, and each part gets five holes. The motor is used to advance the part by 1 inch for each hole. So, you have a repeat loop with a count of 5. Along comes a bad part, detected after the 3rd hole is drilled. You exit the loop with 2 counts remaining.

If you simply reenter the loop again, the next part will only get two holes drilled into it. What you really want is for the loop to reset itself. For this, you must use the Reset Repeat Loop instruction, as shown above.

**Reset Interrupt Option**
This option is used for re-enabling an interrupt as you exit your interrupt handling routine. It is only used for a Go To type interrupt. For a Call/Return interrupt, use the Return from Interrupt instruction to re-enable.

## *Set Output*

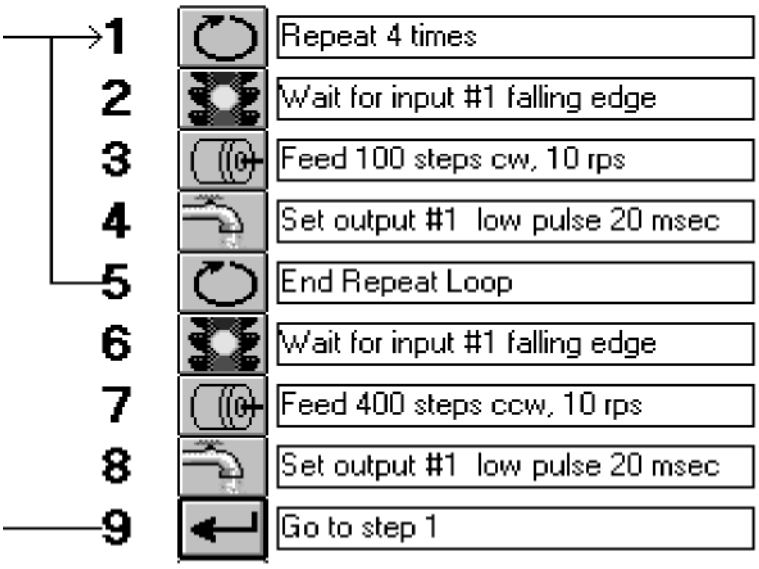



Earlier, we discussed the Wait Input instruction as a way to make the Mis Indexer wait for external events to happen before proceeding with the program. Sometimes you want the opposite: the Mis Indexer should tell other equipment when to proceed. The Set Output command lets you pick one of the three outputs (seven on a BLU-Mis drive) and put a voltage signal on it. For a detailed description of the circuitry and connections, see the section "Wiring Inputs and Outputs" in your hardware manual.

There are four choices of output conditions:
**High** - Makes the photo transistor open. In circuits where the "-" output pin is grounded, and the "+" pin is pulled up, this causes a high voltage to appear on the "+" pin.
**Low** - Makes the photo transistor close. In circuits where the "-" output pin is grounded, this causes a low voltage to appear on the "+" pin.

**High Pulse** - Makes the photo transistor open for a specified amount of time (2 to 500 milliseconds)
**Low Pulse** - Makes the photo transistor close for a specified amount of time (2 to 500 milliseconds)

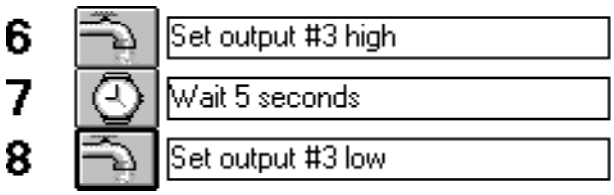
***At power-up, the Mis Indexer sets al 3 programmable outputs high (open circuit).***


1 Repeat 4 times
2 Wait for input #1 falling edge
3 Feed 100 steps cw, 10 rps
4 Set output #1 low pulse 20 msec
5 End Repeat Loop
6 Wait for input #1 falling edge
7 Feed 400 steps ccw, 10 rps
8 Set output #1 low pulse 20 msec
9 Go to step 1

For an example of using the Set Output instruction in your program, let's consider the example of filling containers. Each time the Mis Indexer moves to a new position, it should tell the dispenser that it's arrived. We'll do this with a low pulse, but your choice would depend on the kind of signal the dispenser wants to see in order to be activated.

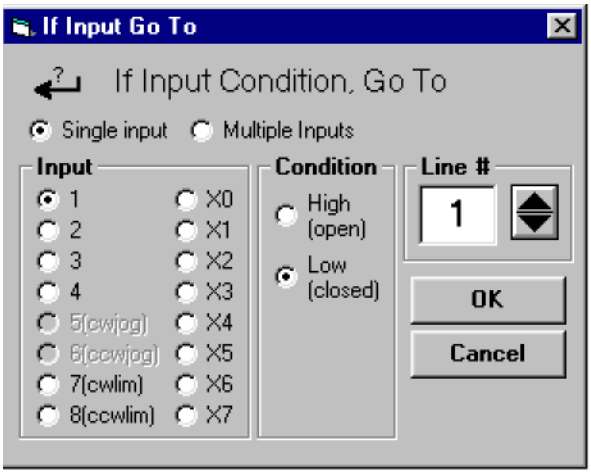
After adding a Set Output instruction after each Feed to Length, we have the program shown on the left.

There may be occasions where you want a long pulse. This can be done by combining two Set Output commands with a Wait Time. The instructions shown below will produce a high pulse of 5 seconds on Output 3.



## *If Input Go To*

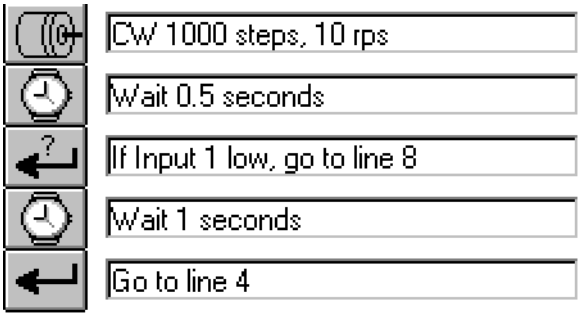
This instruction allows the Mis Indexer to make decisions based on input signals. You can choose “Single Input” or “Multiple Inputs.”




You’ll need to choose an input terminal for the instruction to check. You also need to tell the indexer what signal condition to look for. Finally, you must set the line number that the instruction will jump to if the input condition occurs.

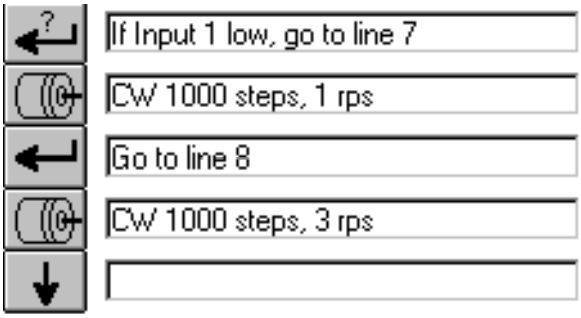
We included the If Input instruction for three reasons.

1) It allows you to skip part of your program based on an external condition. For example, let’s say you are building a machine and the Mis indexer’s task is to feed parts. Normally the indexer waits only a half second before feeding the next part, because that’s how long it takes for your saw to cut the part. But some-times you process parts of a different material that takes longer to cut (aluminum vs. steel, maybe). On the days you run steel parts, you’d like to be able to flip a switch and change the delay between parts to 1.5 seconds. This is how you do it:



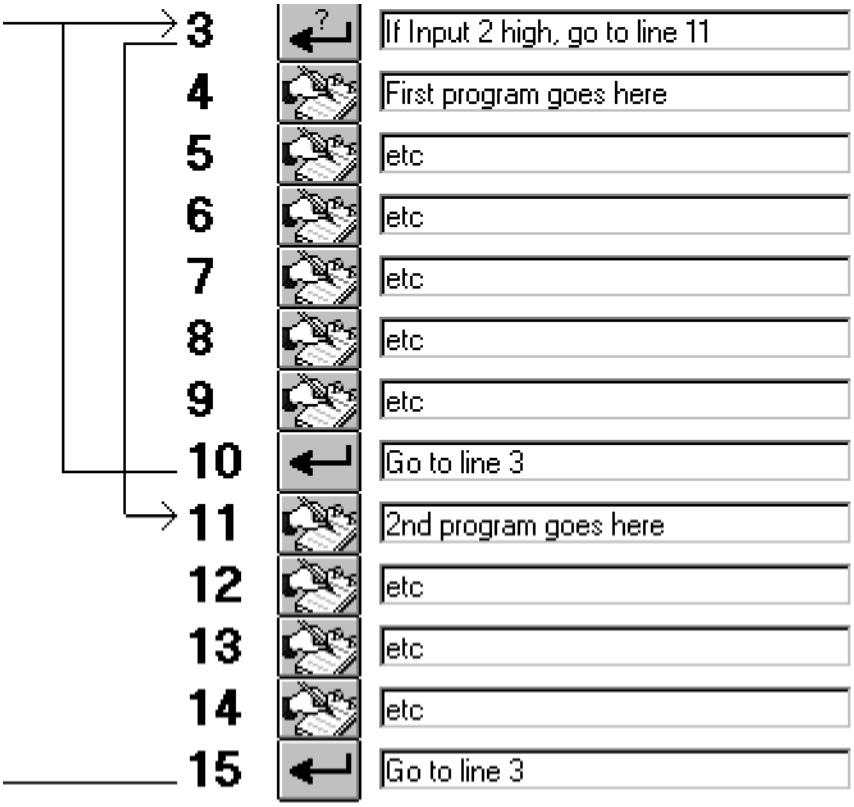

The program feeds a part during line 4. Line 5 makes it wait a half second. If the switch, which you’ve connected to Input 1, is closed (low voltage signal state) then the program jumps to Step 8, skipping the extra 1 second of delay. If the switch is open, the delay occurs. You could then mark your switch’s open circuit position as “Steel” and the closed position as “Aluminum”.

2) The second reason for including the If Input nstruction is to allow you to change a parameter such as distance or speed based on an input. Consider that last example. How will the cutoff saw know how fast to go when I set the switch for “Aluminum” or “Steel?” Well, we could wire the switch that controls the movement of the saw to one of the general purpose inputs. This time we write the program as shown below.




When the indexer gets to Step 4, it will look at the “Steel/Aluminum” switch. If the signal is low (Aluminum) it jumps to the Feed to Length program at Step 7, which moves the saw at 3 revolutions per second. If the switch is high, the program does not jump, but instead executes the Feed to Length at Step 5, which feeds at 1.0 rev/ sec. Then, the Mis Indexer jumps past the second Feed because of the Go To instruction in Step 5. (Don’t forget the Go To or you could end up moving the saw twice.)

3) The final reason we’ve given you the power of If Input is so you can have multiple programs within your 100 line program space. Perhaps what you want your system to do is two completely different things depending on an input. Lets say that each of these tasks requires 4 instructions. This is what you do:




Depending on the state of Input 2, the program will either execute lines 4 - 9 or lines 11 - 15. Either way, the program ultimately returns to line 3 to check the condition of the switch again.

If you select “Multiple Inputs” then you can combine multiple inputs in a logical “or” or “and” expression. For example: “If input 2 low or 3 low or 4 high goto line x”.

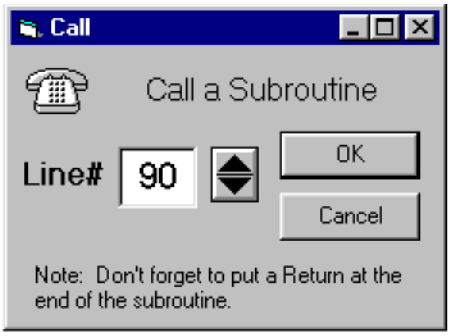
If you have a BLU-Mis drive, you will see more input choices, including X0, the encoder index. With Applied Motion motors, X0 reads high when you are on the index and low everywhere else.

***Note: If you use an If Input instruction to exit a Repeat Loop, the loop does not automaticaly reset the next time you enter it. If you exit a loop by using an If Input instruction, you should use a Reset Repeat Loop instruction to reset the loop. Otherwise, the loop count resumes where it left off.***

## *Call and Return*

Call and Return instructions are used to create and use subroutines. Subroutines are useful programming tools for two reasons. First, if there is set of instructions that appear in your program more than once, you can save program lines by creating a subroutine and calling it from each place the repeated program lines occur. For example, you may have a “handshaking” routine at the end of each move. You could put the handshaking routines near the end of your program, followed by a Return instruction. Then you just Call that subroutine after each move. That frees up program lines so that you can do more.

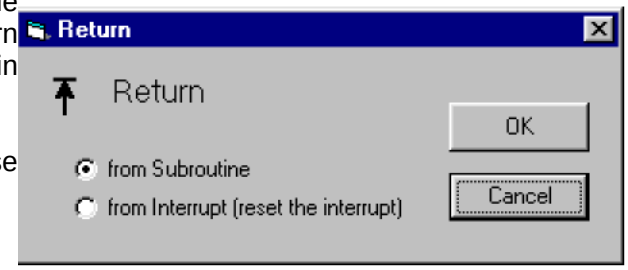
Subroutines can also help you avoid programming mistakes. If you have the same sequence of instruction at more than one place in your program, and you decide to change one of the instructions, you must remember to change all occurrences of that instruction in your program. If the sequence is in a subroutine, you only need to change it in one place.




Example: Suppose that you are feeding parts and then cutting them. After each move you must signal another device to perform the cut, then wait for it to be done. A typical sequence would be:

Set Output (to tell the another drive to make the cutting move)
Wait Time (to give the other device time to respond to the Set Output) Wait Input (wait for other device to signal completion)

To make this a subroutine, just place the three lines near the end of your program (line 90, for example) and add a Return from Subroutine instruction. Each time you make a move in your program, add a Call Line 90 instruction.

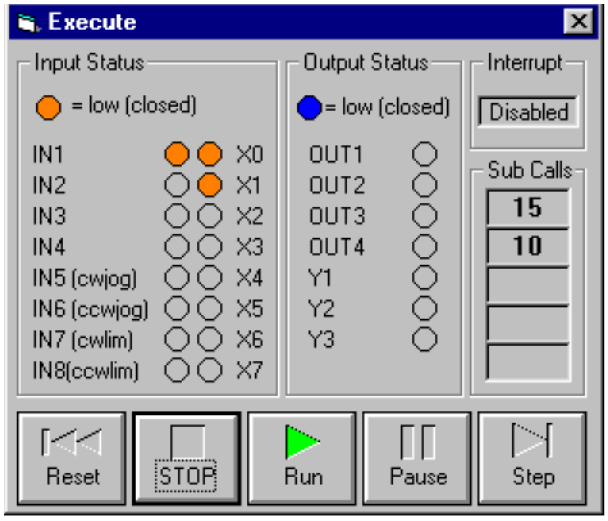



When you program the Return instruction, be sure to choose the “from Subroutine” option.

**Call Stack**

Each time you call a subroutine, the drive must make note of where you came from, so that the subroutine’s Return instruction knows where to go. We store this “return address” on the Call Stack. If you call a subroutine and it calls another subroutine, another value will be pushed onto the stack. This process cannot continue forever, as the call stack is just five levels deep.

If you call a sub that calls a sub that calls another sub, you’re okay, But don’t allow more than five calls without a returns. If you are using interrupts, then you must save one stack level for that, as the interrupt uses one stack level.




**Animation**

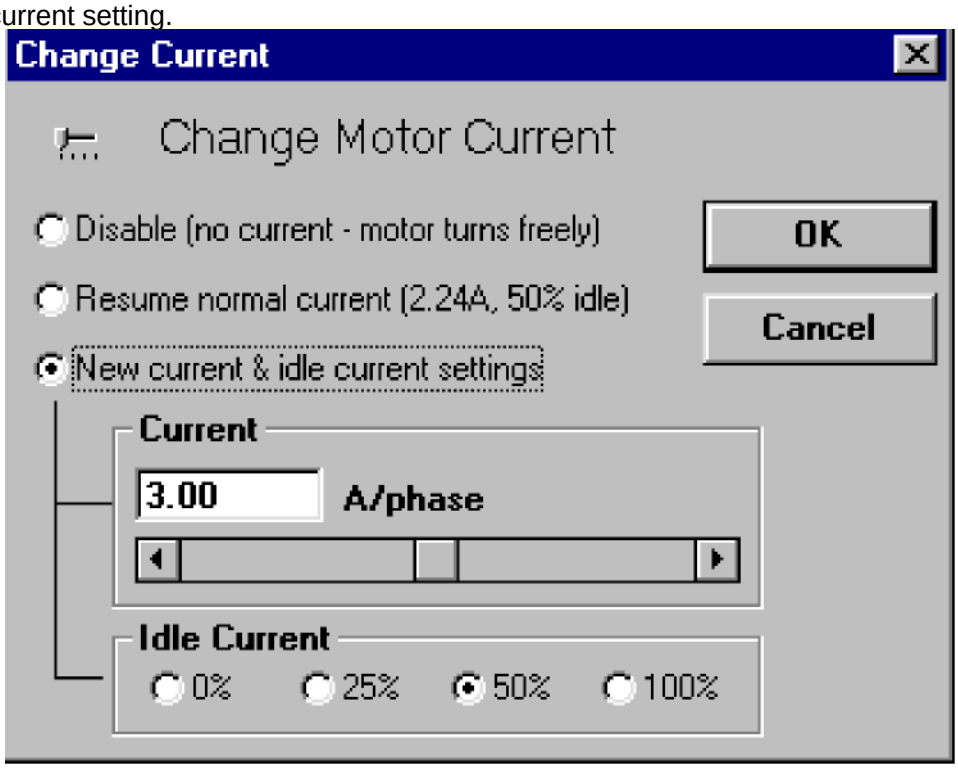
When you are in development mode (drive still being controlled by the Mis Programmer application on your PC) we animate the call stack in the status window. This will give you a feel for how the stack works and whether you’re getting close to a stack overflow.

### *Change Current*

Motor current is normally set on the main screen of the Mis Programmer and applies to all program instructions. But what if you want to change the current on command? For example, you may want to temporarily turn off the motor current while the operator makes manual adjustments to a mechanism, or loads a new roll of labels. Overcoming the holding torque of a step motor by hand can be difficult, so it's sometimes best to shut off the current completely.

In other cases, you may want to temporarily increase the motor current to achieve more torque, but are unable to leave it that way all the time without overheating.

The Change Current instruction allows you to turn off the current, to resume the normal current setting, or to specify a new current setting.



## *Change Servo*

*This instruction is only available if you are using a BLU-Mis servo drive.*

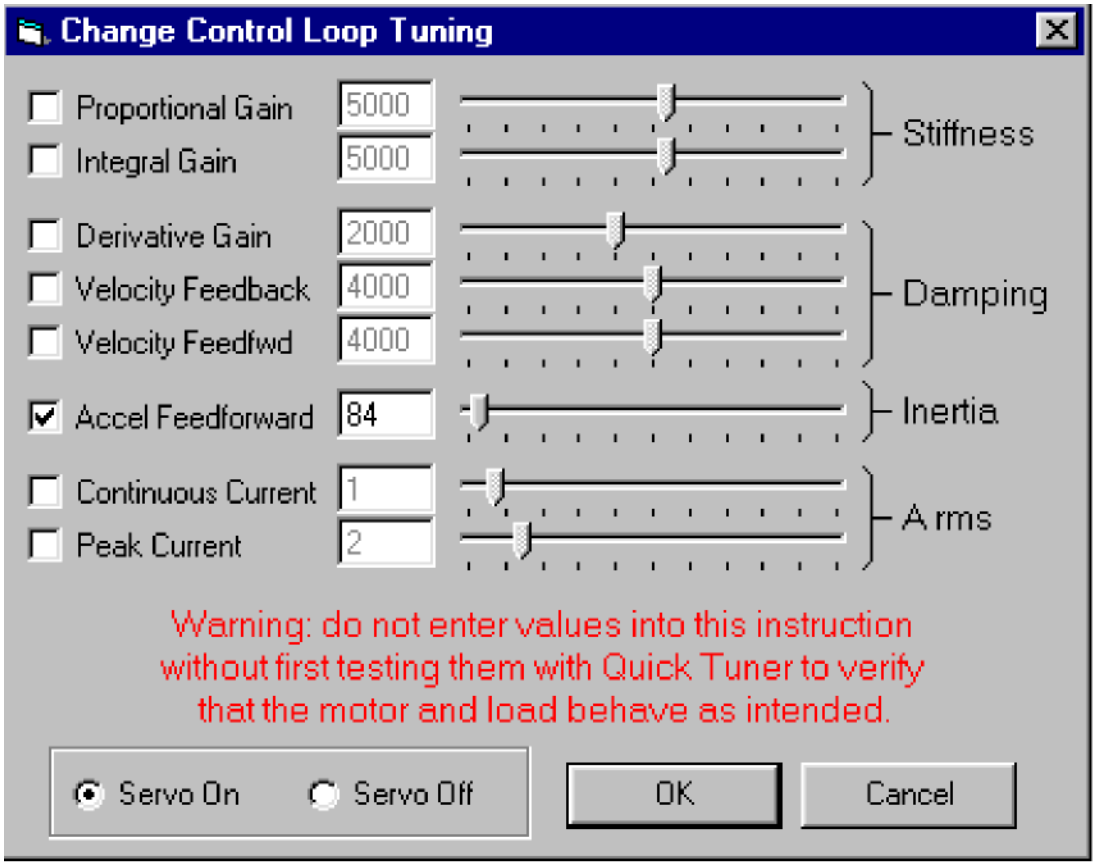
The Change Servo instruction allows you to change the six control loop gains and the continuous and peak current settings. You can also turn the servo loop on or off.

Normally, you will use the Quick Tuner software to configure and tune your drive. However, if you have a changing load you may need to change some or all of the tuning parameters between moves. For example, if you are transporting objects on a linear slide and the mass greatly changes when an object is placed on the slide, you will get better performance if you adjust the gains between moves. In particular, you might want to increase KAFF, the acceleration feed forward, to compensate for increased load inertia.

Before entering values into the Change Servo instruction, you should try these values in Quick Tuner. That way you can measure and observe the performance.

Changing the current will affect the motor torque.

Turning the servo off is useful if you want the motor to spin freely.



### *Comment*

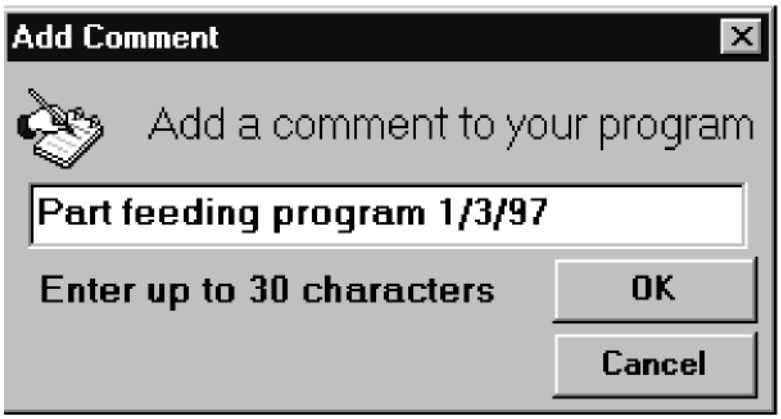
The Comment lets you leave notes in your program. That way if someone else needs to modify your program in the future they'll understand what you've done. The comments can also help you organize the program for yourself.

Whether you save your program to disk or just download it to the drive, the comments stay with it. They do not affect the way your program runs: when the Mis Indexer executes a program, it skips over the comments.

We suggest that you place a comment on the first line of your program to let future programmers know who wrote the program, when it was written, and what it does. You can also put in comments at other places in your program. Perhaps just before a Feed to Length move, you would want to add the comment "Rotate the part 1/4 turn ."

A little time spent now commenting your program might save you a great deal more time later on, when you've forgotten that "Feed to Length 3000 Steps" means "Rotate the part 1/4 turn."

There is a limit to the number of comments your program can have. The Mis Indexer has a "string pool" of 400 characters. All MMI Prompts text goes into the string pool, as any Comment instructions whose strings exceed 12 characters.



## Command Buttons

### Download, Upload & Execute

The Mis Indexer was designed to operate without a host computer once your program is finished and tested. At first, though, you'll probably find yourself running it from the PC much of the time. That way you can quickly make changes in your program to fix errors or conduct experiments.

The Mis Programmer software provides four command buttons for interacting with the indexer-drive.

**Download** transfers the program from the Windows software to the Mis Indexer that's plugged into your serial port. The transfer takes about 3 seconds. You must download the program before you can execute it.

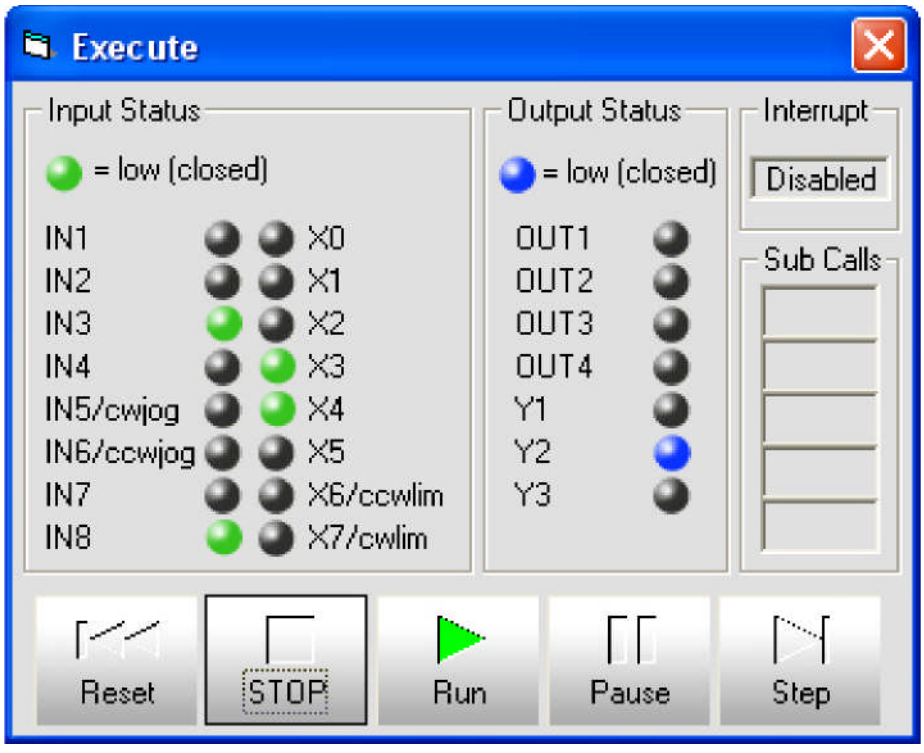
***Note: when downloading to the indexer/drive, make sure the JOG inputs are not activated. If in doubt, remove the JOG CW and JOG CCW connector plug.***

**Upload** lets you extract whatever program is in the Mis Indexer memory and display it on the screen. If you want to modify a program already in your Mis Indexer, you can use the Upload command to bring it back to the PC.

**Execute** tells the Mis Indexer to begin running the program that is in its internal memory, starting on line 1. After hitting the Execute button, you'll see a box appear with a status display and five program control buttons. (Unless your indexer/drive has firmware prior to 1 .40, in which case you'll see a much simpler execute box. Older drives are not able to send real time status information to the PC, and cannot respond to advanced commands like Pause and Single Step.)

If the drive is not connected to the PC or the Mis Programmer software is not running, the drive execute the most recently downloaded program, starting on line 1.

Pressing **STOP** will interrupt the indexer-drive at any point in the program and close the execute panel. You will find this feature useful when the drive starts doing things you didn't intend for it to do.



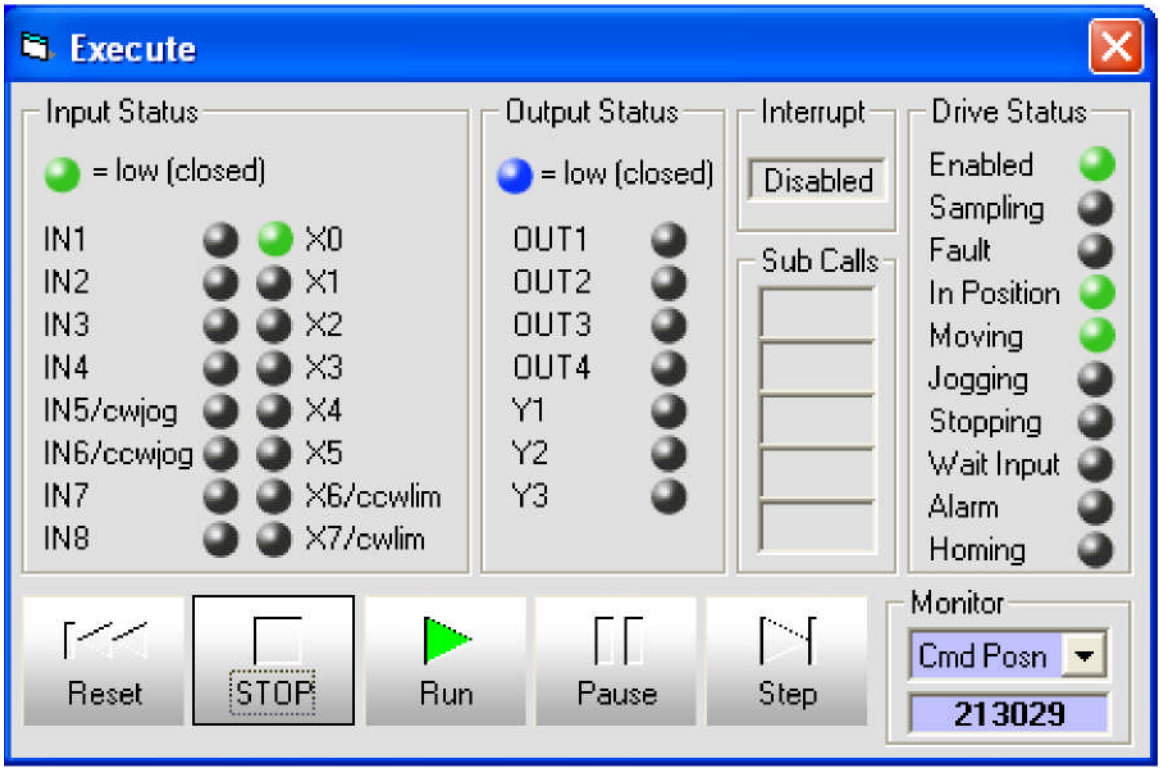

**Pause** halts the program, but does not close the execute box. While the program is paused, the display of inputs is continuously updated, so you can adjust sensors and switches and see the result in real time.

**Step** executes the next line of the program, then automatically pauses it again.

Pressing **Run** makes the program run again, from where you paused it. Reset sends program execution back to line 1.

***If you have drive firmware 2.22 or later and Mis Programmer 2.5.13 or later, the I/O status is real time when your program is running and when it is paused.***

The STAC6-Mis, BLuAC5-Mis and BLuDC-Mis drives with firmware 2.22 or later provide a real time view of the drive's status flags, as shown below. You can also monitor one internal variable of your choice, including motor position, encoder position, position error, motor speed, current, drive temperature, supply voltage and alarm code. This feature requires Mis Programmer 2.5.13 or later.



### *Save, Open, Print & Quit*

In addition to exchanging programs with the Mis Indexer, the programming software can also save & load programs using your hard drive, and can print hard copies of programs using your printer.

The **Save** button lets you save a program to the hard drive. A file dialog box will ask you to pick a name for the program. You can enter up to 8 characters, not including the suffix ".SI5". If you don't enter a suffix, the software will add ".SI5" to your file name automatically. If you type a different suffix, it will automatically change to ".SI5". The characters in the filename must conform to the usual DOS/Windows 3.1 rules. The safest approach is to use only letters and numbers in your filename, and to avoid special characters like "?" or "\".

The **Open** button provides you with a dialog box showing all the ".SI5" files on your drive. Click to select one, then click OK to load it.

Several example programs are installed with your programming software. It's a good idea to load some of the examples and look at them: they may help you with your own application.

**Print** lets you make a hard copy of your program on any printer that's attached to your computer and installed in Windows. Print uses the standard Windows printer dialog, allowing you to specify which printer to use if you have more than one.

If you call for help, we may ask you to print your program and fax it to us. If you have a fax modem that acts as a printer, you can use it to fax the program directly to us from the programming software. You'll have to enter our fax number at some point, which is (831) 761-6544.

The **Quit** button exits the Mis Programmer.

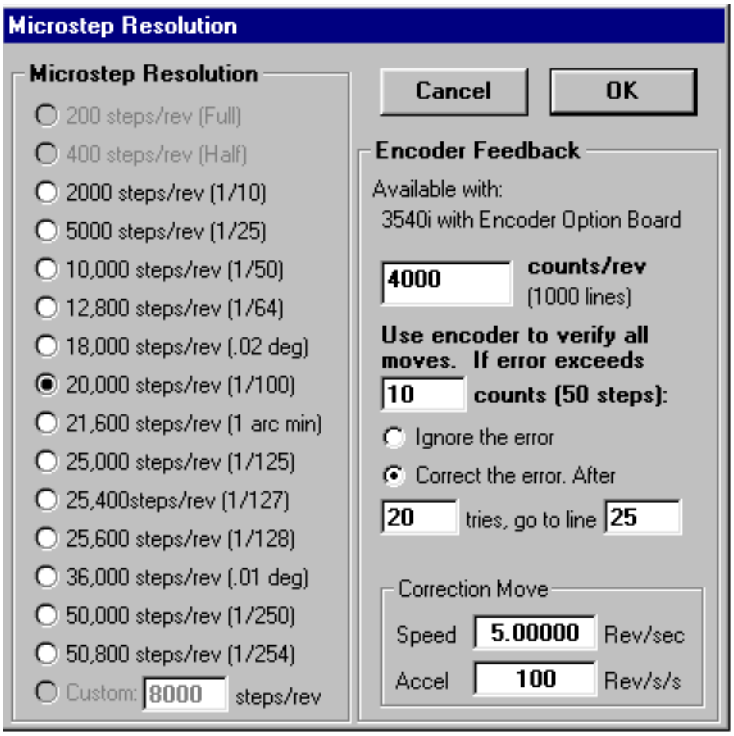
## *Encoder Feedback: 3540i with Encoder Option Board*

Most step motor applications run “open loop,” where a motor position is commanded and the controller assumes that the motor obeyed. If the system is designed with a reasonable amount of torque margin, and nothing blocks the motor’s path, this works perfectly.

However, some applications require feedback from an encoder to verify that the commanded move has been executed correctly. For example, if you are processing very expensive parts, it’s worth spending a little extra money on your motion control hardware so that you don’t destroy the parts if something goes wrong. In other cases, it is desirable to get more accurate positioning by using encoder feedback to compensate for slight errors in the motor position and load linkage.

To support such applications, the 3540i drive that can accept quadrature encoder input with the help of the 1000-175 encoder option board. We can also provide a motor with an encoder mounted on the back.

If you select the 3540i drive and then open the “steps/rev” dialog, you can tell the Mis Programmer software what kind of encoder you have and what to do about any errors that occur.

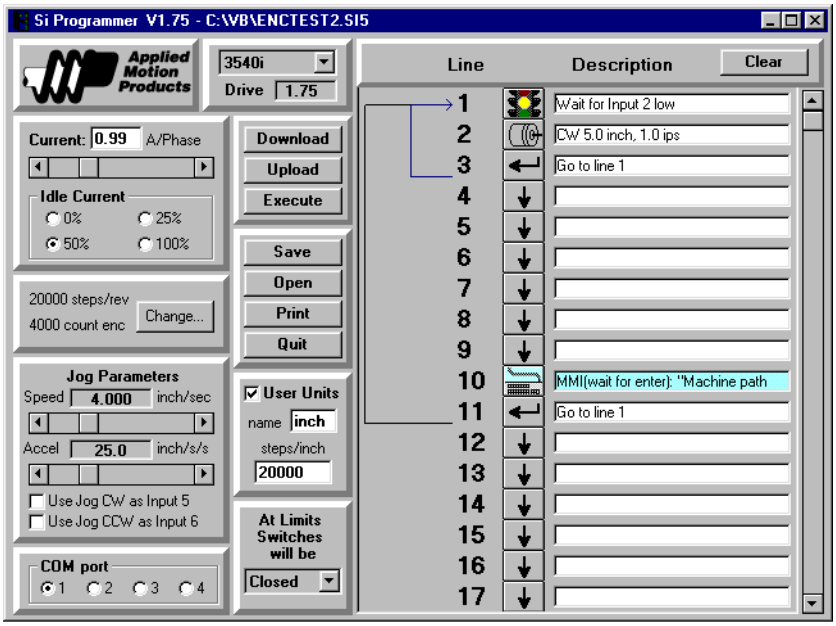



The first thing you should do is enter the counts/rev of the encoder. Many encoders are specified in “lines”. Since the Mis drives use X4 quadrature decoding, the counts/rev will always be 4 times the number of lines. For example, the popular U.S. Digital model E2-1000-250-H encoder has 1000 lines, so you would enter 4000 counts.

***You can use any encoder you want as long as your selection of motor steps/rev is evenly divisible by the counts/rev. In the dialog on the left, the system is set for 20,000 steps/rev and 4000 counts/rev. 20,000 / 4000 = 5, a whole number, so this combination is okay. You could also have selected 36,000 steps/rev with this encoder, but not 12,800.***

Next, enter the maximum amount of position error your application can tolerate. We used 10 encoder counts, which is .0025 inches in our example. If, after a Feed to Length, Feed and Return or Feed to Position instruction, the error exceeds .0025 inches, the controller will automatically try to correct it by making a series of short moves. The controller also performs “static position maintenance” during Wait Input and MMI Prompt instructions. Static position maintenance corrects for external forces that push the motor out of position when it’s supposed to be standing still. In the case of a total obstruction or insurmountable force, it may be impossible to correct the error. In that case, you can have the controller “give up” and branch to another part of your program. That way, you can have an “error recovery” routine. The picture on left shows a simple example of an error recovery routine.

We chose line 10 as the encoder error branch line by entering it in the steps/rev dialog. The error recovery line is automatically shown in light blue so that it's easy to identify when looking at your program. (If you have the printed version of this manual, it probably looks gray.)




In our example, if a position error occurs, the controller tries to correct it up to 20 times, then gives up and jumps to line 10. On line 10, we have an MMI Prompt telling the operator to fix the problem.

**Corrective move speed and acceleration rate**

For firmware version 2.08 and later, you can specify the speed and acceleration rate to be used by the corrective move in the microstep/encoder dialog. These values are not used by older firmware versions - those drives will use the same speed and acceleration rate as the move that failed.

Once the error branch takes place, the autocorrection process is turned off until you branch again by using a Go To or If Input instruction. Thus, the controller will stop trying to make corrective moves while in your error recovery routine, and will automatically resume once you've fixed the problem and jumped back.

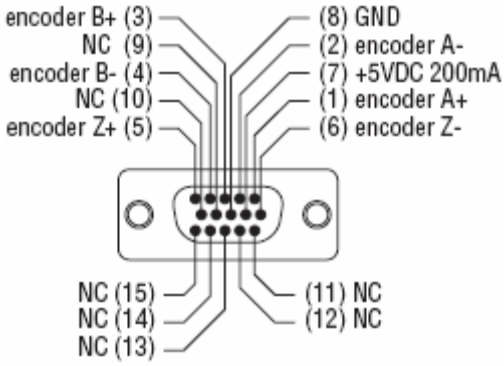
If you have a 3540i drive and you aren't using an encoder, select the option "Ignore the error." That will turn off the autocorrection and error branching. If you are using the encoder, select "Correct the error".

***Notes:***

***1: We do not recommend the use of "0% idle current reduction" if you are using encoder feedback.***

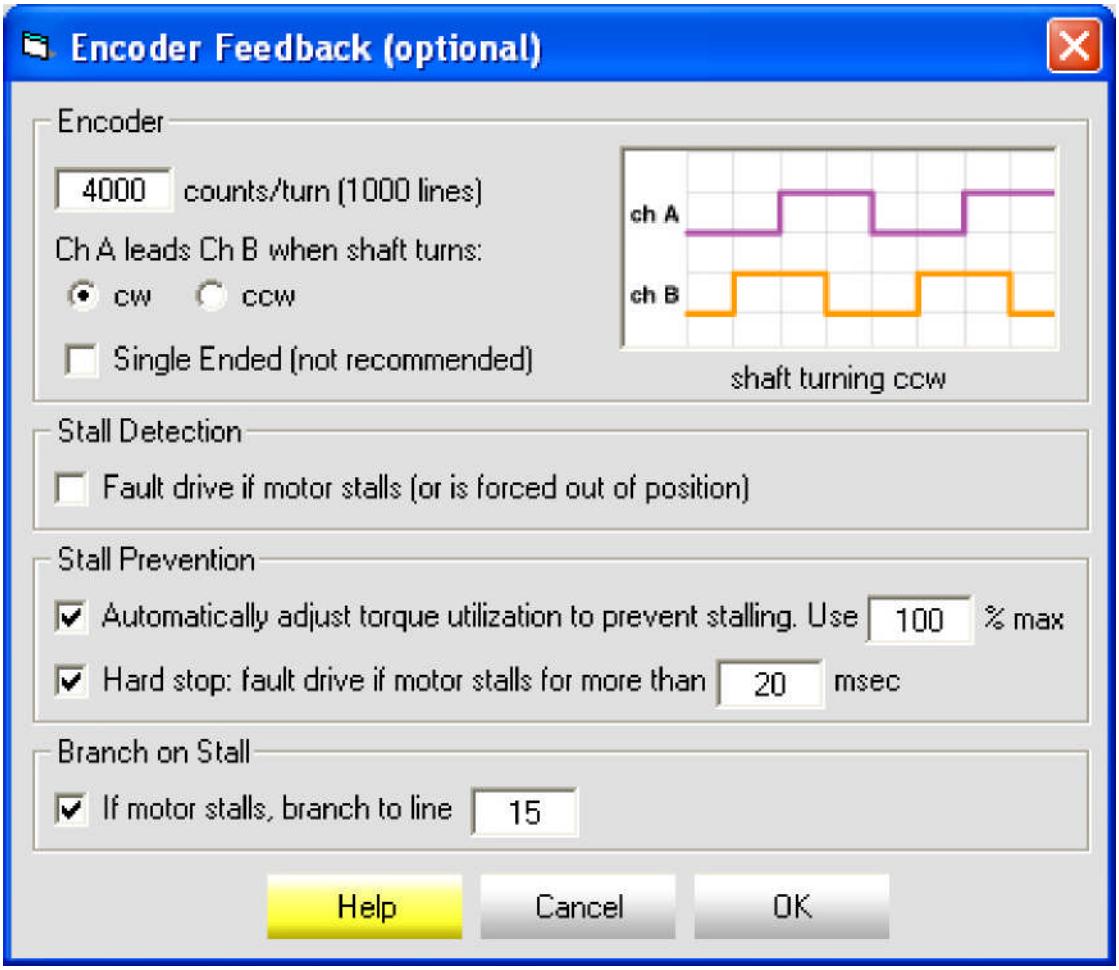
***2: To use the encoder feedback feature, you must have Mis Programmer software version 1.75 or later and drive firmware version 1.75 or later.***

## *Encoder Feedback: STAC6*




The optional encoder connects to the STAC6 using an HD1 5 male connector. You'll have to add this connector yourself according to the diagram below. It is not essential to connect the Z (index) channel.

If you are using an encoder with single ended outputs, shame on you. Differential connections are far less sensitive to electrical interference and life is too short to waste time deciphering the bizarre problems that can occur with a poor quality encoder. That said, single ended encoders should be connected to the A+ and B+ terminals. Leave A- and B- unconnected. They are internally biased to the proper voltage for best results. You'll also need to select the "single ended" box in the encoder dialog or the drive will think you have a broken encoder wire. That's another good reason to use a differential encoder: the STAC6 can detect and broken wire or bad signal and alert you to the problem.




**Encoder Feedback Options**

***Stal Detection*** continuously compares the actual motor position, as reported by the encoder, against the theoretical motor position. If the motor strays so far out of position that it can produce no torque, a position fault occurs. This includes a motor at rest being driven out of position by an external force.

Check the box below Stall Detection if you want the drive to fault when the motor stalls. This fault can be reported by the Fault output and cleared by the Alarm Reset input (see I/O Dialog).

***Stal Prevention*** can prevent many stalls before they occur. The STAC6 achieves this by using the encoder to monitor the lead angle of the motor, a measure of torque utilization. If the motion profile begins to demand more torque than the motor can produce, the velocity is automatically reduced before the motor stalls. To engage stall prevention, simply check the Stall Prevention box and enter the maximum torque utilization you want to allow.
In the event that the motor cannot move at all, such as hitting a hard stop, you can may want to fault the drive after a given amount of time by checking the Hard Stop box and entering a time limit.

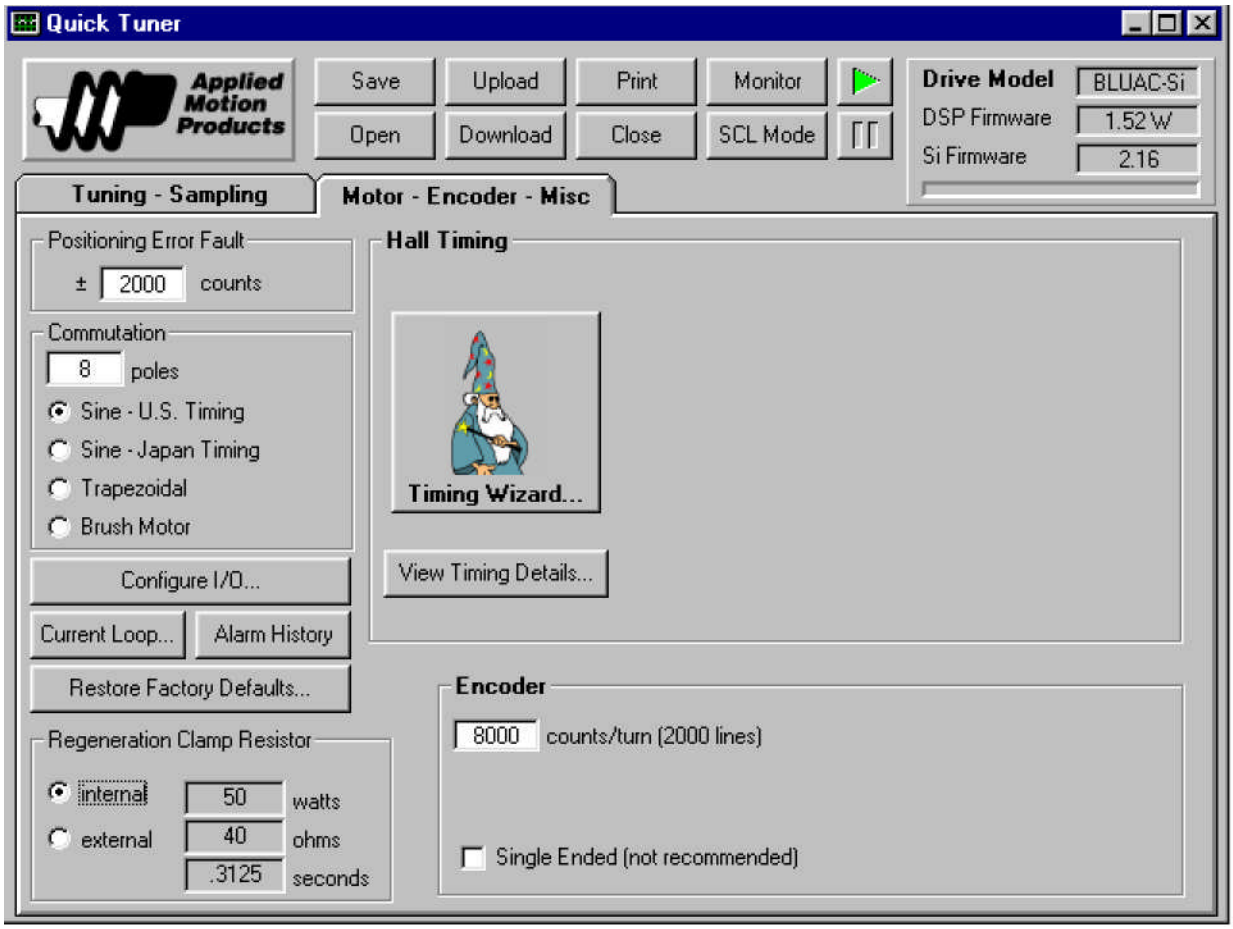
***Branch on Stall*** allows you to handle a stalled motor in your program. Typically you would take some kind of action such as alerting an operator via the MMI or setting an output to signal the fault to another piece of equipment. Once the condition has been handled, you'll need to clear the fault and reenergize the motor with a Change Current instruction (choose the "resume normal current" option). The next time your program branches with a Go To or If Input instruction, the encoder functionality is restored.

# Servo Tuning

If you are programming a BL or BLU servo drive, you can configure and tune the servo system from Mis Programmer. Once you've chosen the correct drive model from the drop-down list, a Servo Tuning button will appear on the left side of the main screen. Click on it to launch the Tuning screen.

The Tuning screen has two tabs: "Tuning - Sampling" and "Motor - Encoder - Misc."

## The Motor - Encoder Tab




This tab is for configuring the motor feedback (encoder and commutation settings), choosing the position error limit and, in the case of the BLU-AC drive, specifying the regeneration resistor. From this panel you can also access the alarm history, I/O configuration and current loop tuner. You can even restore all drive settings to the factory defaults with the click of a button.

### Encoder and Hall Timing - BLU Drives

***If you are using an Applied Motion Products' Alpha, N or M series motor, the factory drive settings are correct for this type of motor. You do not need to enter the encoder resolution or run the Timing Wizard. You can skip this section.***

Three phase brushless DC motors provide three signals that tell the drive when to switch from one phase combination to another. These are called commutation or Hall signals. The relationship of the commutation signals to the motor phases (called "Hall timing") is not the same for all manufacturers. That's okay, because he Applied Motion drives can accept motors with nearly any hall timing.

***A few motors have unequaly spaced Hal signals. These are sometimes caled 60° motors. (Evenly spaced hal waveforms are 120° apart.) The one type of motor you cannot use is one with 60 degree timing. Unless its Hal outputs are differential, in which case you can convert it to 120° by swapping the "-" and "+" outputs of Hal 2. If you have a 60° motor with single ended Hal outputs, you cannot use that motor with the BL or BLu series drives.***

To set the hall timing, you must first wire your system. That includes connecting the motor to the drive and the drive to a power source. Refer to the hardware manual for your drive when making these connections. Once you have connected the motor to the drive, we can configure the timing using the Quick Tuner.

If you are using a motor from another manufacturer, try the Hall Timing Wizard first. In most cases, it can automatically detect your motor timing pattern and configure the Quick Tuner software for it. You may also manually configure your motor by skipping ahead to the section "Manual Configuration."

Hall timing varies among motor manufacturers. The timing diagrams supplied with motors (when they are supplied) differ in their format, too, complicating the task of configuring a motor for the first time. To ease this burden, the Quick Tuner includes a Hall Timing Wizard that automatically detects the necessary configuration for your motor. To use the Hall Timing Wizard, you must do the following:

***Note:- When using BLu servo drives the encoder resolution and the number of poles must be entered before the folowing procedure is attempted.***

1. Connect the motor to the drive. Wire the phases, hall and encoder signals "straight across" as detailed in your drive's Hardware Manual.
2. Do not connect any load to the motor. It is important that the motor be unloaded and completely free to move during the operation of the Hall Timing Wizard.
3. Connect the programming cable to your PC.
4. Launch the Quick Tuner software.
5. Connect and apply power to the drive.
6. In the motor-encoder-drive panel, click on the "Timing Wizard" button.
7. The Wizard will remind you to disconnect the motor from any load.
8. The Wizard will ask you to rotate the motor shaft in the clockwise direction. The Wizard will tell you when to stop turning the shaft.
9. Click OK and the Wizard will do the rest. If successful, the Wizard will determine timing apply those settings to the Quick Tuner set up. You also wil need to tune the drive - we don't have a wizard for that.

## *Encoder and Hall Timing - BL Drives*

***If you are using an Applied Motion Products Alpha motor, the default drive settings are correct for this type of motor. You can skip this section.***

Three phase brushless DC motors provide three signals that tell the drive when to switch from one phase combination to another. These are called commutation or Hall signals. The relationship of the commutation signals to the motor phases (called "Hall timing") is not the same for all manufacturers. That's okay, because the Applied Motion drives can accept motors with nearly any hall timing.

***A few motors have unequaly spaced Hal signals. These are sometimes caled 60° motors. (Evenly spaced hal waveforms are 120° apart.) The one type of motor you cannot use is one with 60 degree timing. Unless its Hal outputs are differential, in which case you can convert it to 120° by connecting the "-" output of Hal 2 to the drive, and the "+" outputs of Hals 1 and 3. If you have a 60° motor with single ended Hal outputs, you cannot use that motor with the BL or BLu series drives.***

To set the hall timing, you must first wire your system. That includes connecting the motor to the drive and the drive to a power source. Refer to the hardware manual for your drive when making these connections. Once

you have connected the motor to the drive, we can configure the timing using the Quick Tuner.

If you are using a motor from another manufacturer, try the Hall Timing Wizard first. In most cases, it can automatically detect your motor timing pattern and configure the Quick Tuner software for it. You may also manually configure your motor by skipping ahead to the section “Manual Configuration.”

Hall timing varies among motor manufacturers. The timing diagrams supplied with motors (when they are supplied) differ in their format, too, complicating the task of configuring a motor for the first time. To ease this burden, the Quick Tuner includes a Hall Timing Wizard that automatically detects the necessary configuration for your motor. To use the Hall Timing Wizard, you must do the following:

1. Connect the motor to the drive. Wire the phases, hall and encoder signals “straight across” as detailed in your drive’s Hardware Manual.
2. Do not connect any load to the motor. It is important that the motor be unloaded and completely free to move during the operation of the Hall Timing Wizard.
3. Connect the programming cable to your PC.
4. Launch the Quick Tuner software.
5. Connect and apply power to the drive.
6. In the motor-encoder-drive panel, click on the “Timing Wizard” button.
7. The Wizard will remind you to disconnect the motor from any load.
8. The Wizard will ask you to rotate the motor shaft one turn in the clockwise direction. (See the sketch.) Because the drive has no absolute frame of reference, you must provide one by turning it clockwise. The amount of rotation does not have to be exactly one turn, but try to get it within 30 degrees.
9. Click OK and the Wizard will do the rest. If successful, the Wizard will determine hall timing and the encoder direction and apply those settings to the Quick Tuner set up. ***You still must enter the exact encoder counts per turn manually. You also wil need to tune the drive - we don’t have a wizard for that.***

If the Hall Timing Wizard fails, you might still be able to configure the system manually. But some motors cannot be used with the BLu drives.

## *Manual Timing Configuration (BL Series only)*

1) Locate the timing diagram that was supplied with your motor. If you don’t have one, proceed to option (2). If you have a timing diagram, compare it to the one shown in the Hall Timing frame. If it doesn’t look the same, try selecting different timing patterns until you find one that matches.

Please note that the names may differ. We label the halls 1, 2 and 3. Our phases are called A, B and C. But not everyone is as smart as we are, so you may find that your motor maker has chosen U, V and W or R, S and T. It really doesn’t matter as long as you remember how you connected the motor to the drive. Just substitute your motor’s nomenclature for ours.

If the relationship of the Halls to the motor phases is inverted, click on the check box marked “inverted.”
2) Sometimes it is best to just try things until something works. For this reason, we have included a Test button.

Before you try this, make sure that the motor is not connected to any load, and that it is firmly secured. Once we find the right Hall timing, the motor is going to run continuously, very fast.

With the drive powered up, and motor connected, click on the Test button. If the motor does not move, try another setting from the list until it runs smoothly.

3) Next, stop the motor by pressing the Stop button (the Test button alternates between Stop and Test each time you press it). Click on “inverted”, then press Test again. Does the motor run the same speed as it did in step 2 (as indicated by the RPM readout below the Test button)? If not, go back to Step 2 and find another timing pattern that works equally well in both directions.

4) Okay, we've almost got it. When you hit the Test button, which way does the motor run? If it goes counter-clockwise (when looking at the front of the motor), you are done. Save your set up to disk before you forget your results.

5) If the motor was running cw, then change the "inverted setting." It should then run ccw. Save your results.

## Encoder Direction (BL Series only)

***Note: the Hal Timing Wizard sets the encoder direction automaticaly. If you used the Timing Wizard to configure your motor, you can skip this section.***

Incremental encoders provide two signals to the drive: channel A and channel B. The drive determines the direction of motion from the relationship of channel A to channel B.

For all Applied Motion motors, channel A leads B when going ccw. If you are using another motor, this may not be the case. Documentation is sometimes sketchy on this issue, so it's often best to just try both choices.

Once you are certain that your hall timing is good, try to turn the motor shaft with your fingers. If it comes back, the encoder direction is correct. If not, then the motor will run continuously at a fairly high speed.

***Note: if you've been using the Test button to help set the Hal Timing, then the servo loop may be off. If the shaft turns freely, the servo loop is off. Cycle the power on your drive to turn the servo back on, then repeat the experiment.***

## Positioning Error Fault

Positioning error is the difference, in encoder counts, between the actual position and the commanded position of the motor. A small amount of positioning error is a normal part of a servo system. But sometimes the unexpected can happen. A wire might break, a sensor could fail or the motor may encounter a physical obstruction. You might even one day forget to set up and tune a drive before installing it into a system. In all of these cases, you'll want to know that something is wrong as soon as possible and without damaging anything. For this reason, the BL and BLu drives include a position error fault limit. Anytime the position error (as reported by the encoder) exceeds this limit, the drive cuts power to the motor and enters fault mode. (See the Servo Faults section for details.)

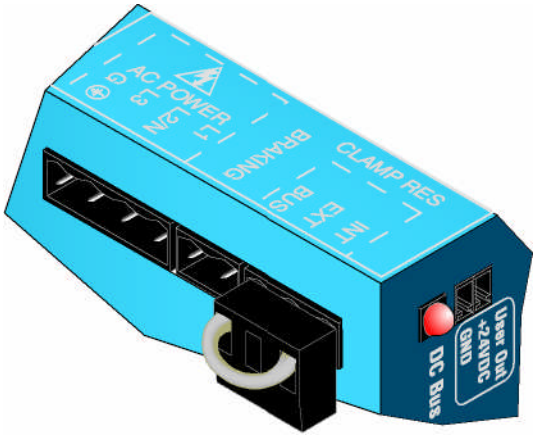
You can set the fault limit to as little as 10 encoder counts, or as much as 32000. When you're first tuning the system, you should set this value high so that the drive doesn't shut down as you experiment with tuning parameters. Once the drive is properly tuned and you know how much error to expect during normal operation, you can set an appropriate fault limit. For example: set the Quick Tuner's scope to plot position error. Execute some aggressive sample moves, using the maximum speed and acceleration that you plan to use in your application. If the maximum position error is, say, 50 counts, then you could safely set the fault limit at 100.

## Regeneration Clamp

If you rapidly decelerate a load from a high speed, much of the kinetic energy of that load is transferred back to the power supply. This can trip the overvoltage protection of the power supply, causing it to shut down. For the BL7080i, BL7080x, BLU100 and BLU200 series DC powered drives, we recommend the addition of one RC050 regeneration clamp for each power supply. The BLMis7080 and BLx7080 drives have built-in regen clamps that require no adjustment.

Due to the extended power range of the BLUAC5 drive, the built-in regen clamp may not be enough in all cases. If you experience regen faults while operating your drive, you should first check to see that the internal regen resistor is actually connected. On the bottom of the drive, you should see a jumper wire connecting the

outer two terminals of the regen connector, as shown below. If the wire or connector are missing, remove power from the drive and replace them.




If the internal resistor is connected and you still experience regen faults on your BLU-AC, you'll need to connect a larger resistor externally. You also must inform the drive that you've done this, by selecting the "external" option button. Then you must enter the continuous power capacity, in watts, and the resistance. Now for the hard part: entering the number of seconds that the peak regen power level can be sustained by your external resistor. The peak regen power is 160,000 / R where R is the resistance of your external resistor. For a 50 ohm resistor, the peak power is 3200 watts. The resistor manufacturer can help you with this.

Consult the hardware manual when connecting the external resistor to the drive.

### *Reset Factory Defaults*

If things really get out of hand, you can click this button to return the drive to the way it was when we shipped it from the factory.

### *Alarm History*

This provides the same view of recent alarms and faults as the button on the main Mis Programmer screen. Please see page 21.

### *Configure I/O*

This activates the same dialog as the Configure I/O button on the main Mis Programmer screen. Please refer to page 17.

## *The Tuning - Sampling Tab*

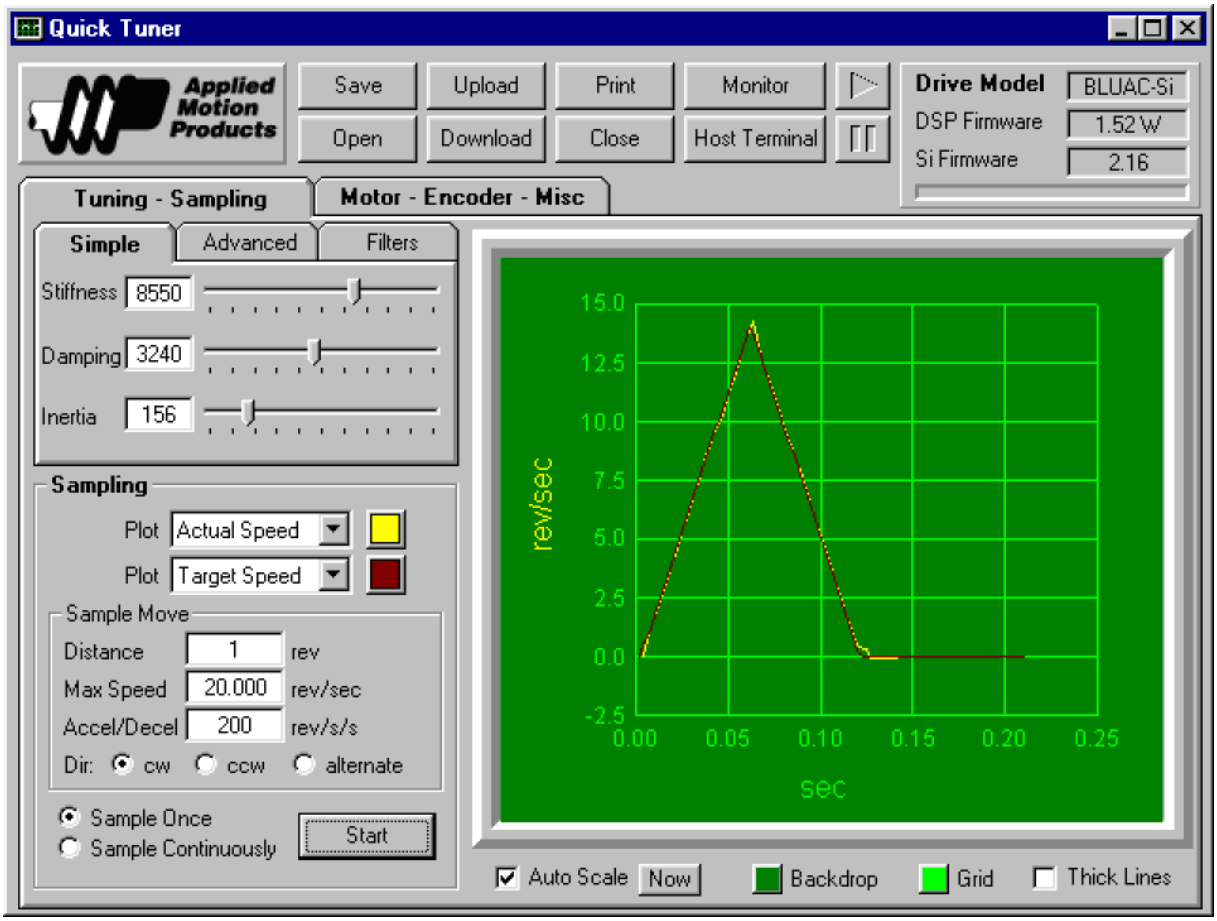
This tab allows you to set the control loop parameters (also called servo gains) and adjust the filters. You can also experiment with the response to your choices of gains and filter settings using a built-in digital scope.

### *The Scope*

At some point, you are going to want to observe how well your choice of gain parameters works. We have included a sampling oscilloscope feature for this reason. All you have to do is select the move profile you'd like to try, by entering a move distance, a maximum speed, and the acceleration rate. You should select a move that is meaningful to your application. There is no point in optimizing your speed around 50 rev/sec if your application only calls for speeds of 10 rps. The same is true for distance and acceleration.

The oscilloscope window can display either one or two measurements at a time. The choices include Actual Speed, Target Speed, Position Error, Current Command and Actual Position. The drive only samples what you have selected, so if you change plot parameters you must sample again to update the display. By making the oscilloscope "full screen" you can get a better display resolution and see your move in more detail.

You may also choose clockwise or counterclockwise rotation. If your load is vertical, you may want to observe the differences. If your load is linear, then you can't keep sampling in the same direction or you'll run out of travel. For applications where travel is limited, choose "alternate direction." Then the drive will change the direction after each move.




If you have selected “Sample Once”, you will get one sample move, and one graph each time you click on “Start.”

The scope automatically scales the graph to fit the data. If you are tuning your drive to minimize position error, you may find it hard to tell “at a glance” that the error has changed because the graphs keep re-scaling to fit the entire scope area. If uncheck the “Auto Scale” box, the graph vertical axes will no longer change and it will be easier to judge the response to changes you’ve made in the tuning parameters. If the trace becomes too large to fit or too small to see in detail, you can click on the “Now” button to perform a onetime re-scaling of the data.

If you don’t like the colors we’ve chosen for the two traces or the grid or background, you can change them by clicking on the color buttons. Don’t worry, we won’t be offended. If you have enlarged the scope by dragging the border of the Quick Tuner screen or clicking the maximize button, you might want to check “Thick Lines”.

Sometimes it is convenient to keep the sampling process running automatically while you adjust the gain parameter slide bars. To do this, select “Sample Continuously.” The drive will start sampling and plotting when you click on “Start” and will not stop until you click “Stop.” (Note: the Start button becomes a Stop button when you click it in Continuous sample mode.)

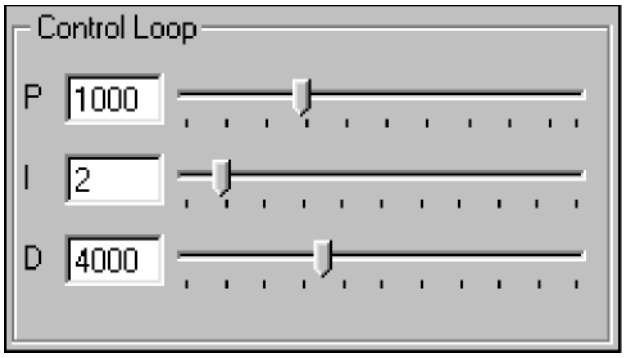
It is not necessary to click on the Download button when sampling. If the Quick Tuner detects any changes in the drive settings, it will automatically download the new settings before each sample move.

Each time the settings are downloaded to the drive, they are written into the nonvolatile memory, and will still be there the next time you power up the drive.

## Control Loop Tuning - BL Drive

Like most modern servo drives, ours employ sophisticated algorithms and electronics for controlling the torque,

velocity and position of the motor and load.




Sensors are used to tell the drive what the motor is doing. That way, the drive can continuously alter the voltage and current applied to the motor until the motor does what you want. This is called “closed loop control.”

One of the loops controls the amount of current in the motor. This circuit requires no adjustment other than specifying the maximum current the motor can handle without overheating.

On the BL7080x, BL7080i, BLMis7080 and BLx7080 drives the position control loop is a Proportional-Integral-Derivative (PID) type. This type of closed loop control is used widely, not just in motion control but also in other process controls. We chose PID control for our drives because it is easy to understand, which makes it easier for you to set up our drives for your application.

The PID loop compares the intended motor position to the actual motor position as reported by the encoder. The difference is called error, and the PID loop acts on this error in three ways: the Proportional term, the Integral term and the Derivative.

### P: The Proportional Term

The simplest part of the PID loop is the proportional, or P, term. The drive applies current to the motor in direct proportion to the error. Here’s an example: if the motor were standing still, and you suddenly turned the shaft by hand, you’d want the drive to increase the motor current so that it goes back into position. The farther you disturb the motor from it’s target position, the more the torque will increase. The P term (also called P gain) governs how much torque you get for a given amount of error Un. In general, if you have more load inertia, you’ll need more torque and therefore a higher P gain.
The torque provided by the P term is T = KPUn.

### I: The Integral

If you think about the previous example for a moment, you may realize that P alone will not give you perfect position. If you applied one ounce-inch of torque to the motor, it would move out of position. The P term will increase the motor torque until it is producing as much torque as you are. Then the motor stops moving. But there is still error. The I term adds up all the error that the drive has seen and produces a torque that is added to the torque command from the P term:

T = KPUn + KIÓ(U).

In our example, the P term allowed the motor to reach equilibrium at a position where the applied torque from your hand equaled the torque of the P term. Thus, the error was not zero. But the I term will keep adding up that error and continue to increase the torque until the motor truly returns to the target position.

### D: The Derivative

So far, we’ve just talked about a motor that is disturbed when standing still. But the objective of motion control is to get that motor moving on it’s own. The problem with electric motors is that they tend to be very “springy”, a condition known as “underdamped”. If you tried to run a motor with a pure PI controller the motor would overreact to small errors, creating ever larger errors, ultimately becoming unstable. If you knew what the

motor was going to do before it did it, you could prevent this. For those of you who studied calculus, you may recall that you can predict what something is going to do by it's rate of change, or derivative.

If you are driving your car into your garage, do you wait until you are in fully in the garage before hitting the brakes? That would be a bad idea for you, your car, and the back wall of the garage. Instead, most people slow down as they see the distance between them and their objective get smaller.

A motor drive can control a motor better if it examines the rate of change of the position error and includes that in its torque calculation. For example, if the motor has error, but the error is decreasing, back off on the torque. That's what the "D" term does. The final, exact PID equation used in the BL7080 and BL7080i is:

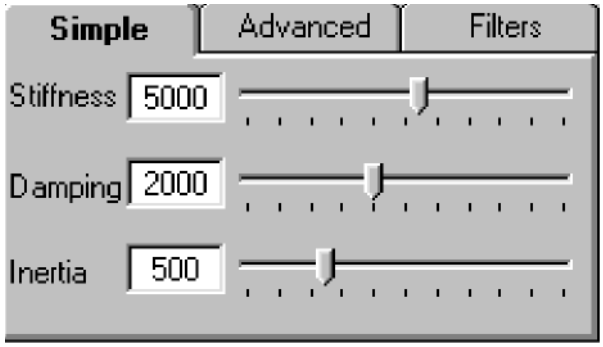
T = KT [KPUn + KIÓ(U) + KD(Un - Un-1)] / 74880

where
Un is the error in encoder counts
Un-1 is the error of the previous sample KT is the torque constant of the motor.

This calculation is performed 4266 times per second.

## *Control Loop Tuning - BLU Drive*

Like most modern servo drives, ours employ sophisticated algorithms and electronics for controlling the torque, velocity and position of the motor and load.



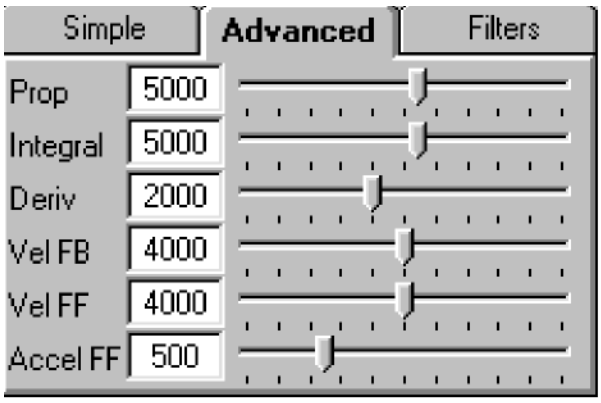

Sensors are used to tell the drive what the motor is doing. That way, the drive can continuously alter the voltage and current applied to the motor until the motor does what you want. This is called "closed loop control."

One of the loops controls the amount of current in the motor. This circuit requires no adjustment other than specifying the maximum current the motor can handle without overheating.

On the BLU servo drives the position control loop is a Proportional-Integral-Derivative (PID) type. This type of closed loop control is used widely, not just in motion control but also in other process controls. We chose PID control for our drives because it is easy to understand, which makes it easier for you to set up our drives for your application.

The PID loop compares the intended motor position to the actual motor position as reported by the encoder. The difference is called error, and the PID loop acts on this error in three ways: the Proportional term, the Integral term and the Derivative.

The BLu series drives add a three additional gain terms to enable greater system control: velocity feedback, velocity feedforward and acceleration feedforward. That's a total of six gain parameters, which can be a little intimidating for a first time user. The Simple tab combines the six parameters in just three: Stiffness, Damping and Inertia. For advanced users, you can click on the Advanced tab and access all six gains separately.




### *Stiffness Gain Terms*

#### *P: The Proportional Term*

The simplest part of the PID loop is the proportional, or P, term. The drive applies current to the motor in direct proportion to the error. Here's an example: if the motor were standing still, and you suddenly turned the shaft by hand, you'd want the drive to increase the motor current so that it goes back into position. The farther you disturb the motor from it's target position, the more the torque will increase. The P term (also called P gain) governs how much torque you get for a given amount of error Un. In general, if you have more load inertia, you'll need more torque and therefore a higher P gain.

The torque provided by the P term is T = KPUn.

### *I: The Integral*

If you think about the previous example for a moment, you may realize that P alone will not give you perfect position. If you applied one ounce-inch of torque to the motor, it would move out of position. The P term will increase the motor torque until it is producing as much torque as you are. Then the motor stops moving. But there is still error. The I term adds up all the error that the drive has seen and produces a torque that is added to the torque command from the P term:

T = KPUn + KIÓ(U).

In our example, the P term allowed the motor to reach equilibrium at a position where the applied torque from your hand equaled the torque of the P term. Thus, the error was not zero. But the I term will keep adding up that error and continue to increase the torque until the motor truly returns to the target position.

## *Damping Gain Terms*

### *D: The Derivative*

So far, we've just talked about a motor that is disturbed when standing still. But the objective of motion control is to get that motor moving on it's own. The problem with electric motors is that they tend to be very "springy", a condition known as "underdamped". If you tried to run a motor with a pure PI controller the motor would overreact to small errors, creating ever larger errors, ultimately becoming unstable. If you knew what the motor was going to do before it did it, you could prevent this. For those of you who studied calculus, you may recall that you can predict what something is going to do by it's rate of change, or derivative.

If you are driving your car into your garage, do you wait until you are in fully in the garage before hitting the brakes? That would be a bad idea for you, your car, and the back wall of the garage. Instead, most people slow down as they see the distance between them and their objective get smaller.

A motor drive can control a motor better if it examines the rate of change of the position error and includes that in its torque calculation. For example, if the motor has error, but the error is decreasing, back off on the torque. That's what the "D" term does. The final, exact PID equation used in the BL7080 and BL7080i is:

T = KT [KPUn + KIÓ(U) + KD(Un - Un-1)] / 74880

where
Un is the error in encoder counts
Un-1 is the error of the previous sample KT is the torque constant of the motor.

This calculation is performed 4266 times per second.

### *Vfb: Velocity Feedback*

As motor power per size has gone up so has the size of the loads. As more performance is asked of the servo motor we may need to add more damping. Velocity feedback has been added to the BLu servo drive to provide greater damping for the larger loads.

This term adds in the motor actual Velocity as negative feedback and usually works in conjunction with the velocity feedforward term (see below). If the velocity of the motor matches what is expected no feedback value is generated. If however the velocity's do not match the negative feedback helps to "damp" the differences in velocity. Typically both terms are set to the same value.

### *Vff: Velocity Feedforward*

The velocity feedforward term works with the velocity feedback term to add more damping capability to the servo algorithm. The feed forward value is generated by the "Trajectory Calculation" algorithm. This setting is useful for minimizing position error while in motion by automatically providing needed torque to drive the load rather than waiting for position error to build up and then compensating. Applications which require the load position to be accurate throughout the move, such as CNC machining, benefit from velocity feedforward.

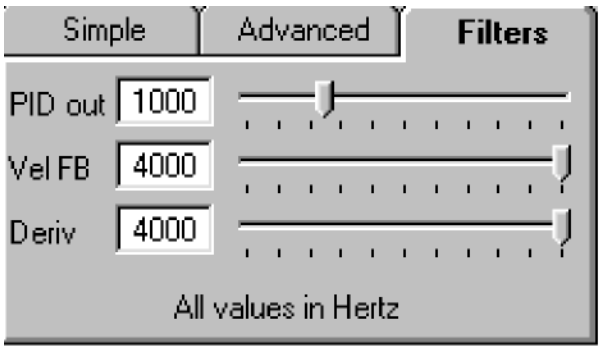
## *Inertia Gain Term: Acceleration Feed forward*

By anticipating the torque needed to accelerate a load, the BLU servo can handle higher inertial loads and provide better positional accuracy during the accel/decel segments of a move. The acceleration feedforward term does this by adding an acceleration value to the control value. The acceleration value is derived from the Trajectory Calculation during the acceleration and deceleration phase. It should be set in direct proportion to the load inertia. The position error display of the scope can help you correctly set the acceleration feedforward.

## *Filters*

The BLU servos provide three control loop filters for special situations. The is the PID Output Filter. If your system is subject to mechanical resonance, below the natural frequency of your system so that the PID output does not

If you have a large inertial load, you'll probably find that you need to set the and I, to get good response. Then you will want to increase the damping to likely to be so tight that if you have a springy, all metal coupling it may "buzz" or "squawk." Reducing the frequency of the velocity feedback and derivative filters can remove this objectionable sound.



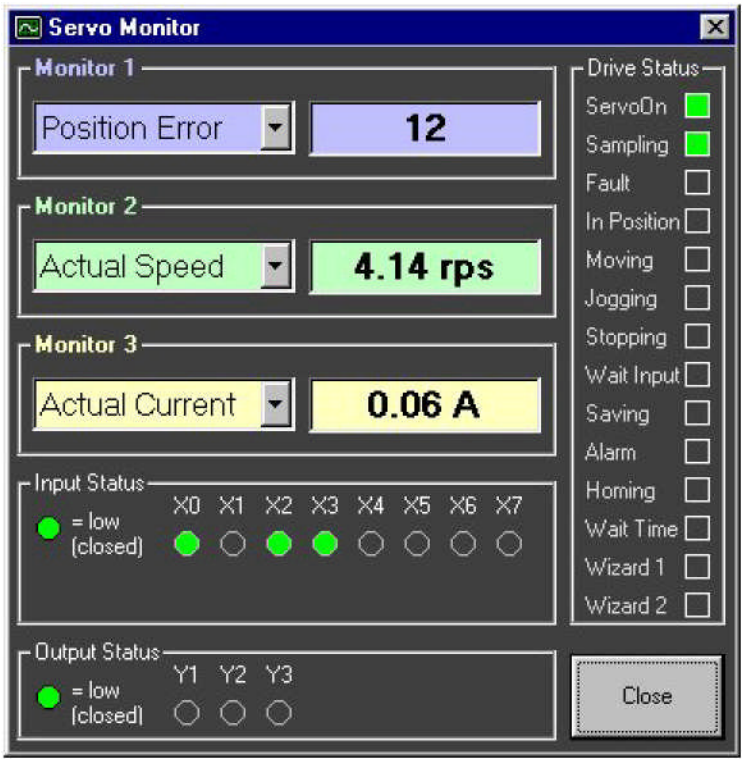

## *Monitor*

One of the buttons near the top of the Quick Tuner screen is labelled Monitor. If you are using a BLu servo drive, you can click on it to reveal the Monitor panel. Designed to assist with commissioning and fault finding this configurable screen will display all the condition of all the I/O and drive status outputs.

The monitor also features three configurable displays that can show live information about the drive. Each can be configured to display one of the following parameters.

| | |
|---|---|
| Position Error | Actual Speed |
| Target Speed | Command Current |
| Actual Current | Supply Voltage |
| Drive Temp | Encoder Count |
| Alarm Code | Ain Voltage |

***Note: By clicking on the Encoder Count, you can return the value to zero.***




### *Host Terminal*

The Host Language is not used with the Mis Programmer. However, in special situations our technical support staff may ask you to bring up the terminal for diagnostic purposes.

## *BL Tuning Tutorial*

Now that we've reviewed the basics of the PID control algorithm and the sampling scope, it's time to learn how to tune a servo system.

Before you begin, make sure that the motor is connected to your load. The whole point of tuning a servo is to get the best PID parameters for your application.

You'll also need to have current, hall timing and encoder resolution correctly specified. If you have not already done this, please do so now.

We've included set up files for all of our motors. If you are using an Applied Motion motor, open the set up file for that motor now. That will correctly set the hall timing, current and encoder settings. It will also give you some good values of KP, KI and KD to start the tuning process.

To locate the correct file for your motor and load, you'll need to know your motor size, the rated voltage and the inertial load of your application.

- For a 400 watt, 60 volt motor with 1:1 inertia, the file name is "A400V60X1 ".
- For a 100 watt, 24 volt motor with 5:1 inertia, the name is "A100V24X5".

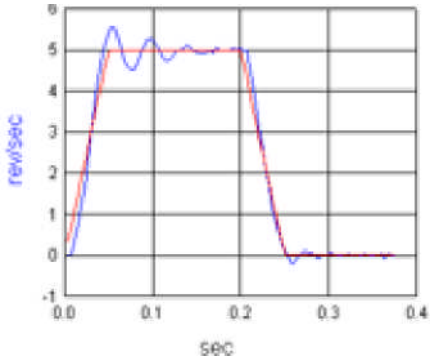



图 1
KP=1 000, KI=5, KD=4000
1 rev, 5 rps, 100 rps$^2$

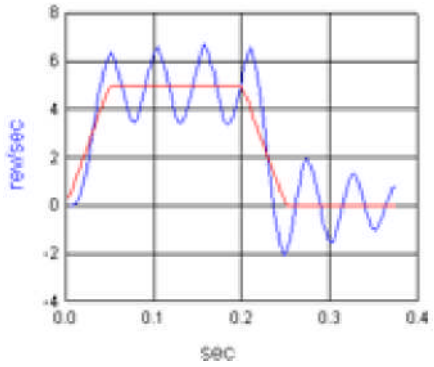



图 2
KP=1 000, KI=5, KD=8000
1 rev, 5 rps, 100 rps$^2$

The easiest parameter to set is KI. For most applications, any value between 1 and 5 will work. For now, leave KI at the default value.

Before adjusting anything, set up a sample move that is meaningful to you. Ultimately, you need to try a sample move where the speed, distance and acceleration are values that you plan to use in your application. Initially, though, don't push it. Keep the acceleration below 200 rev/sec/sec and speed below 10 rev/sec, until we are certain that the system is stable.

We'll also need to specify which parameters we want to look at. For now, leave the settings as they are: actual speed and target speed.

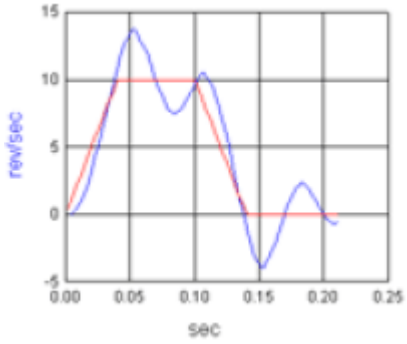



图 3
KP=1 000, KI=5, KD=8000
1 rev, 10 rps, 250 rps$^2$

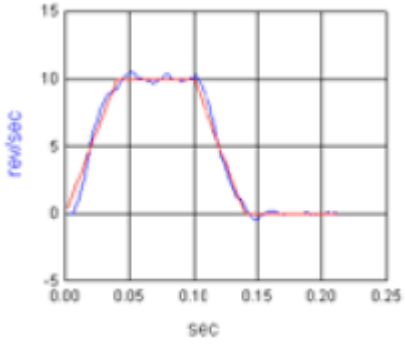



图 4
KP=4000, KI=5, KD=8000
1 rev, 10 rps, 250 rps$^2$

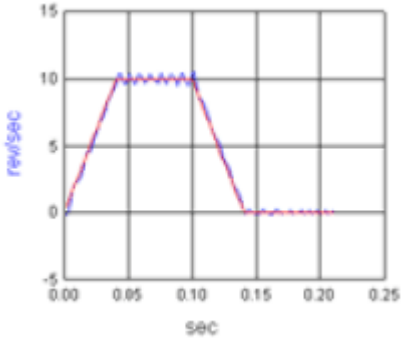



图 5
KP=4000, KI=5,KD=8000
没有负载

Once you've chosen a sample move, click on Start. The motor should move, then the drive will send some data back to the PC and you'll see a plot of speed versus time. No doubt the actual speed won't exactly match the target speed. Our objective is to get it close enough for our needs.

Let's try an example. Trace 1 shows the effects of KP=1 000, KI=5 and KD=4000 on a motor with a load inertia of 744 g-cm2 (4.5 times that of the motor). The motor oscillates about the target speed, never settling in. The system is barely stable. We need more damping, so let's increase D.

Trace 2 shows the effects of D=8000. The motor settles out more quickly now, and is stable for this move. It does overshoot a little when it reaches speed and again when stopping, so something is still wrong. Let's try a more aggressive move. Increase the sample move speed to 10 rev/sec and acceleration to 250. Trace 3 shows the results. The overshoot is large. This time it's not because D is too low, but because P is too low. We don't have enough gain to respond to the demands of a high acceleration rate with high load inertia.

Trace 4 indicates the effect of increasing P to 4000. (Note: I tried several values between 1000 and 4000. It's never a good idea to change a parameter by more than 50% between samples, because once the system goes totally unstable, things can get ugly. You might even break something.)

The increase P gain makes a huge difference in the response of this system. It is now tight and well damped. If our application required higher speeds or acceleration rates it would be a good idea to test their effect now. A little fine tuning could be needed.

At this point you might ask yourself, why can't I use these parameters for any system? Let's try a quick experiment: removing the load inertia. Do you see some small oscillations in Trace 5? They're at a higher frequency than the oscillations we saw before. That's because the problem now is not too little gain, but too much. The system reacts so quickly to position errors that it always overcompensates. This is called dithering. It is a source of position and velocity error and makes an annoying sound.

This is just a small example of what can happen if you make a large change to your system after tuning it.

Despite the fact that we did encounter some dithering when the load was removed, it is still very stable over a load range of 164 to 900 g-cm2. That robustness, combined with its simplicity, make the PID algorithm ideal for servo motor applications.

So far, we've only sampled the target speed and actual speed. You can also look at current, actual position, and position error. Traces 3A and 4A depict position error for the moves shown in Traces 3 and 4. It is no surprise, having already seen the speed ripple in Trace 3, to find high position error, too. The low gain system of Trace 3A has over 150 counts of position error throughout the move. Trace 4A, with higher gain, has only 55 counts of position error during acceleration, and less than 10 counts at speed.

It is informative to view current during a move. If you guessed that the motor doesn't need much current except when starting and stopping, you'd be right. And if you also think that more current is required to accelerate at 250 rev/sec2 than at 100, well, you're right again. In Trace 2B, the peak current is about 1.1 amps. In Trace 4B, where the acceleration rate is 2.5 times higher, the current is just over 3 amps. Since this motor has a 5 amp peak current rating, we could push it a little harder.

Reducing the inertia from 900 g-cm2 to 164 (by removing the load) reduces the current to 0.7 amps, as seen in Trace 5B. Note that the current is "spiky" because the system is reacting too quickly to small errors. In Trace 6B, where the system is better tuned for a light load, the current waveform is smooth.
Once your servo system is properly configured, save your settings to a file, then power down the drive and exit the Quick Tuner software.

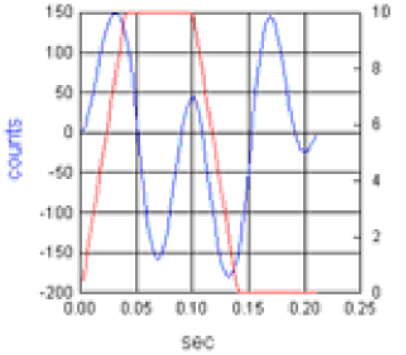



图 3A
位置误差

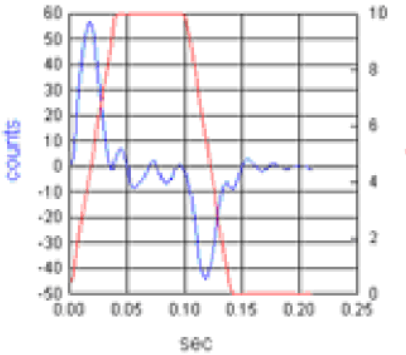



图 4A
位置误差

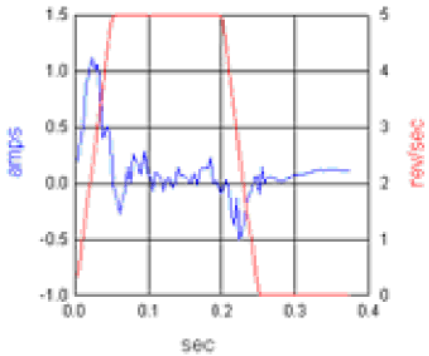



图 2B
电流
KP=1000,KI=5,KD=8000

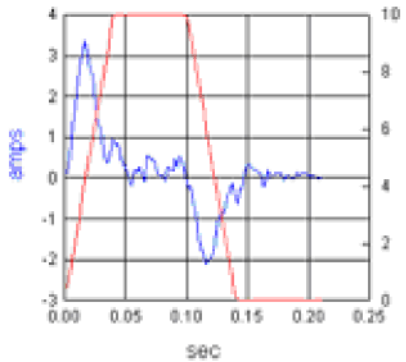



图 4B
电流
KP=4000,KI=5,KD=8000
1rev, 10rev/s, 250 rps$^2$

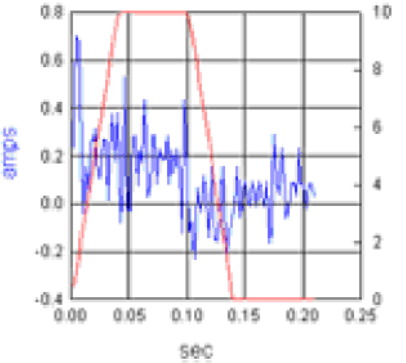



图 5B
电流
KP=4000,KI=5,KD=8000
1rev, 10rev/s, 250 rps$^2$

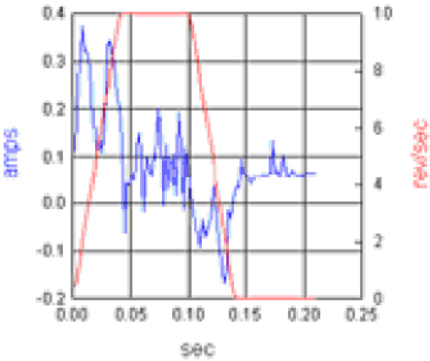



图 6B
电流
KP=1000,KI=5,KD=8000
1rev, 10rev/s, 250rps$^2$

## BLu Servo Tuning Tutorial

Blu Servo drives have the same “PID” control parameters as the BL7080i and BLMis7080 plus they have a few more. As motor power per unit size has increased so has the load requirements for servo motors. The basic PID is great for loads that are not greater then 5:1 inertial mismatch (the difference in inertia of the motor and the load). But, to handle the increased demand 3 more control parameters are added to enhance control of high inertial loads. The three new parameters are Velocity Feedback (KVf), Velocity Feedforward (KVff) and Acceleration Feedforward (KAff). You could say we now have a “PIDVFA” servo control algorithm.

The first thing you may think is “Oh no not more things to adjust and I have no idea what they are”. Well it’s not really that bad. In QuickTuner we have grouped these parameters together in three main functions:

1. Stiffness - KP & KI; These parameters work primarily to keep the servo position with minimal error through-out the move profile.

2. Damping - KD, KVf & KVff; These work to minimize oscillations and overshoot in the motion profile. The KD works mainly on the higher frequency oscillations while the other two work at lower frequencies especially with high inertia loads.

3. Inertia - KAff; This parameter is specifically used to counter the affects of large inertial loads during acceleration and deceleration.

We have made it so that you can adjust the servo parameters in groups or with the “Advanced” button individually. When tuning the BLu servo’s we start out in the same way as the BL’s. We can actually get pretty good control with just the “KP” and “KD” terms.

As with the BL servo drives we have provided “Files” that contain a good starting point for tuning the Servo drive with our Alpha Servo motors. We have expanded the range of inertia’s and now have files for each motor that cover inertia’s of 1:1, 1:5 and 1:10.

### Getting Ready for Tuning

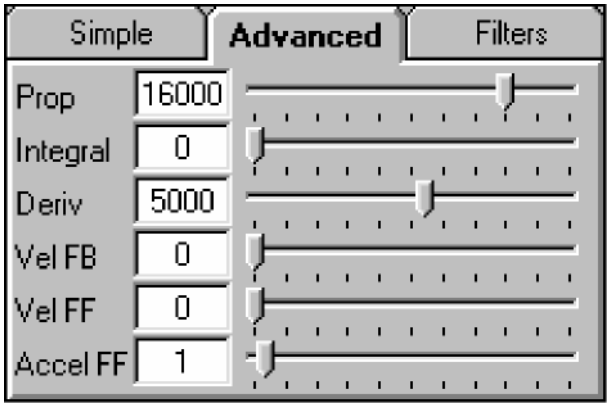
Before you begin it’s good to do some homework on the load the motor is intended to drive. The two primary load issues to consider are the “Frictional” and “Inertial” torque requirements. In a servo system “Frictional Torque” is the easiest to deal with so primarily it is important to know (or estimate) the Inertial Torque. Knowing the Inertial Torque requirement will aid later in know how much gain will be needed in some of the tuning parameters. When calculating or estimating the inertial load you will come up with a number that is in units such as “g-cm2” or “oz-in-sec2”. These are good units to use because AMP Alpha Servo motors are rated using them. For AMP Alpha Servo motors specifications refer to the product catalog or visit out Web-site.

With a good estimate of the Inertial load and knowing the inertia of the motor you are using you can now select a file from a list generated at the factory that best represents your load. You will need to do a little more math to come up with an Inertia mismatch ratio. The file names list the motor first then the load. The files are available typically in 1:1, 1:5 and 1:10 ratios. Select the nearest one and open it. QuickTuner will now be loaded with parameters that will give you a start at tuning the servo system.

We need to give QuickTuner a little more information before we test the servo system. The seemingly obvious things to know are the Max Speed, Acceleration and Distance requirements of the sample move. But what is the not so obvious is what profile shape is required to operate the load properly. The motor may be able to accelerate the load very quickly but may induce significant “Ring” in the motion profile. It might be better to accelerate slower and go to a higher Velocity to minimize the ringing. Deciding the best profile for a given move is sometimes more “Art” then hard calculation.

With all this said, you will have to make a good first guess at the motion parameters to begin the tuning pro-

cess. QuickTuner has a sampling Oscilloscope that will allow us to execute a move and display a variety of measurements.




### 1) Entering a Sample Move

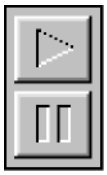
Enter move profile values in the “Sample Move” section. For this example, we will try a move distance of 8 revolutions. We need to choose speeds and acceleration rates that represent the actual application. For this exercise try 30rps and 200 rev/s/s.

### 2) Start with the KP & KD parameters

Before we try a move let’s only start with the Proportional (KP) & Derivative (KD) gain parameters. Record the values of the other parameter settings and “Zero” out there values. Do this in the “Advance Control Loop” screen by clicking the “Advanced” tab. The “Accel Feedforward” can only be set as low as “1 ”.

Starting with only these two terms is a good safe way to begin. They are the minimum required in a servo system.

***Note: If things go wrong (they usualy do) there is a “Play/Pause” button in the upper right corner of the QuickTuner Window under the label “Servo”. Clicking the “Pause” wil disable the servo, clicking “Play” wil enable the servo. Be ready to click the “Pause” if things don’t go wel.***



### 3) Let’s Plot a Move

Start by selecting “Actual Speed” and “Position Error” for the “Plot” selections. Make sure the direction is set correctly, in some cases you may want to select alternate to avoid running the mechanism into a hard stop. For now select the “Sample Once” button.

Click the “Start” button and observe the results. It may not look very good as shown in the next figure.

Other problems may have occurred during the move depending on how things were set up. If a “Fault” occurred you will have been asked to clear it, but the drive will be left disabled until the “Play” button is clicked.

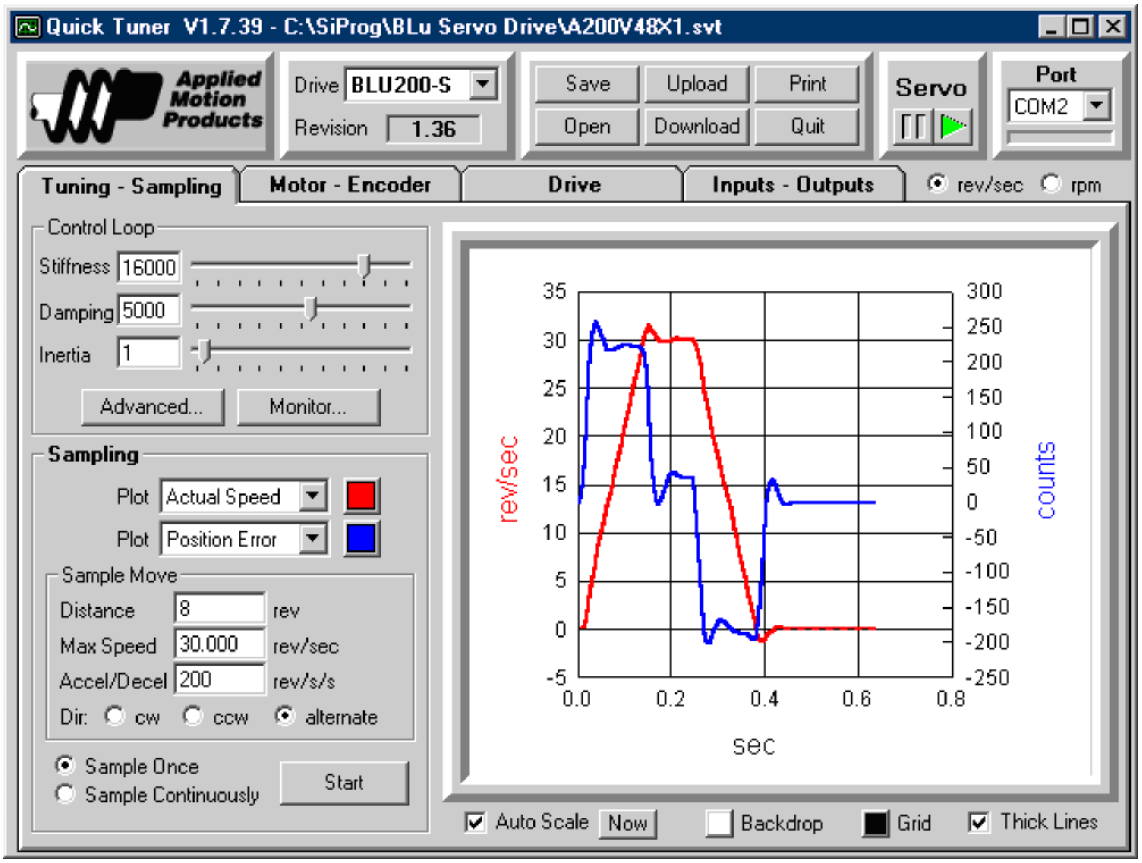
***NOTE: Clicking the “Pause” button then the “Play” button clears a fault and enables the drive.***

4) Now we will adjust the motion parameters to get the desired move profile. You can repeat the move by clicking the “Start” button. If the drive continues to fault you may be exceeding “Current limit” or the “Position Limit”. These can be set in the “Motor -Encoder” and “Drive” tabs.

To see what current is being required of the drive select “Current” in one of the “Plot” lists and click “Start” again to observe a move. This will show the current profile during the move and may give a clue as to why a “Fault” is occurring.

After a successful move is accomplished you can begin to do a little tuning. Adjust the KP and KD parameters and observe the results. Be careful with the KD parameter: too little gain and the system will oscillate. Too much gain may cause the system to “squeal” from a high frequency oscillation.

In some cases where a very “springy” coupler is used between the motor and load, the KD parameter may need to be reduced until the system is stable.




At this point don't worry to much about the larger "Position Error". We will take care of that later.

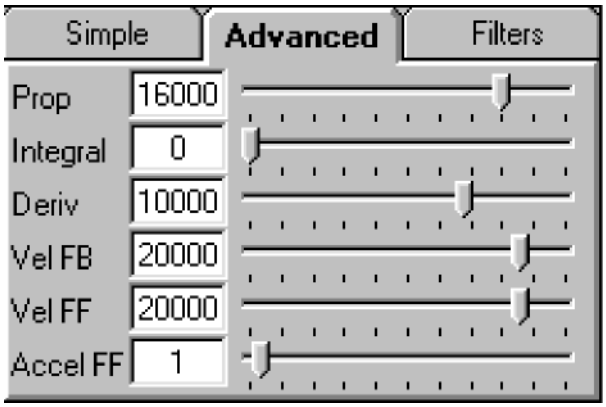
### *5) The return of KVf and KVff parameters*

Now add back in the Velocity Feedback (KVf) and Velocity Feedforward (KVff) parameters that were zeroed in step 2. For large inertial loads these values can be also be large.

The goal with these terms is to minimize the "Overshoot" and get rid of the "Ringing" when accelerating and decelerating. In the plot above, the red trace shows overshoot when the maximum speed is reached, and again when the motor reaches zero speed. There is no ringing in this plot, so we are almost damped enough.

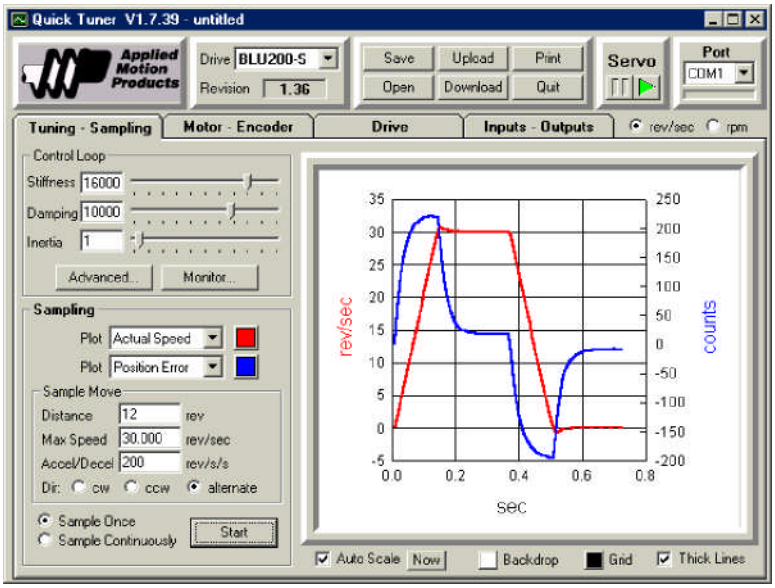
We don't need to eliminate all the "Overshoot" at this point because we have another term that will help.

Typically the KVf and KVff are kept the same. If we want to adjust all the "Damping" values together at the same time use the "Damping" slider on the main QuickTuner Window. This control keeps the damping values ratioed to each other when adjusting the gains.

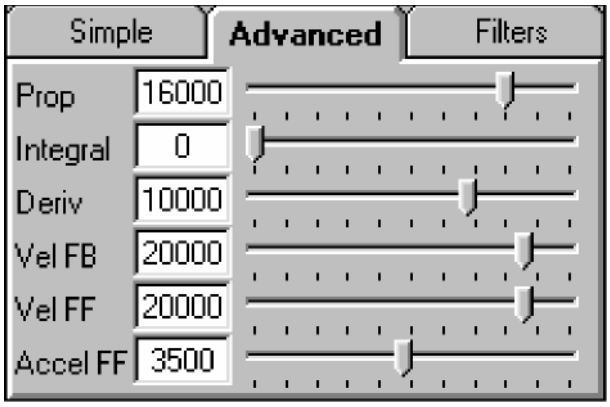


To much gain on the “Velocity” parameters may cause an oscillation, usually apparent when running at the Max Speed or when Stopped. Reduce the gain until the oscillation is acceptable.

As a general guideline the Velocity Gains values are typically 2X the Derivative Gain value.




## *6) Adding in the KAff parameter*



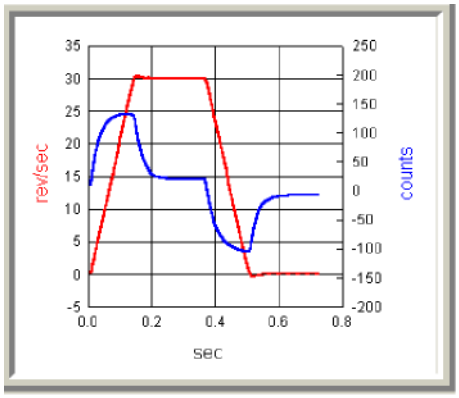

The Acceleration Feedforward (KAff) is a different kind of term from what we have just been working with. This term is used to deal with the inertia of a system. It will request more current during the acceleration and deceleration phases of the move profile.

Start by adding in 1/2 of the recorded value from the “file”. In order to visually see the effect this will have on Position Error make sure the Autoscale Check Box at the bottom of the window is cleared. Now click the “Start” button and observe the results.

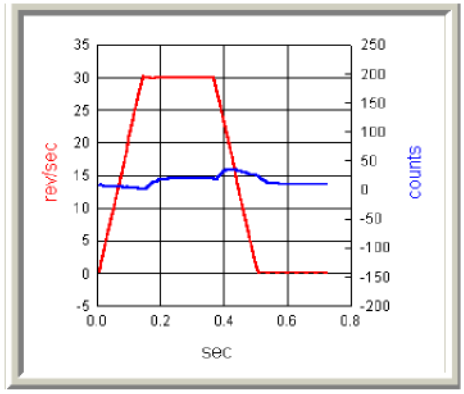
You should notice a reduction in the Position Error (the peak values). The KAff term has a somewhat proportional affect on the Position error. If the error was reduced to half by setting KAff to 3500, then doubling it to 7000 should eliminate the rest of the error. If not you can estimate the next setting. Divide the previous error by the difference in error value then multiply this times the KAff value.

KAff = previous error/delta error * KAff
KAff = 220/90 * 3500
New KAff = 8555

Remember this is an estimation. If after doing this the Position Error goes negative during acceleration, we went too far. Adjust the value in smaller amounts to get as near zero error as possible. At any time you can click the “Now” button near the “Auto Scale” to zoom in on the new Position Error value.




**KAff = 3500**



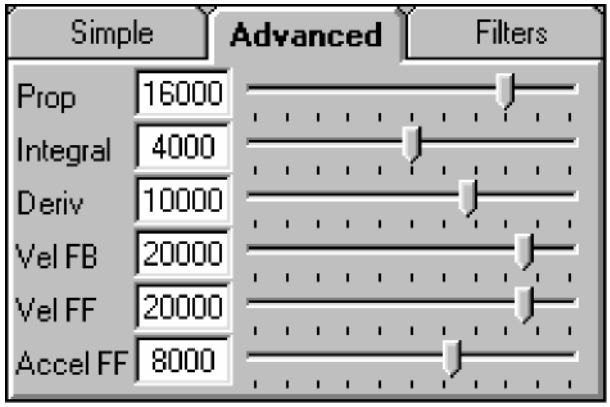

**KAff = 8555**

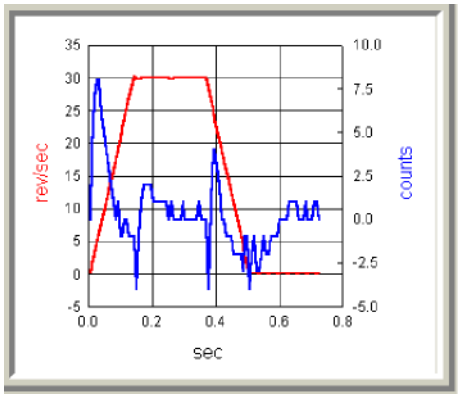
You may have also noticed that the over-shoot of the Max Speed and Stopping is now reduced. This is because the servo control is having to do less work to maintain good control.

### *7) Finishing off with the KI parameter*

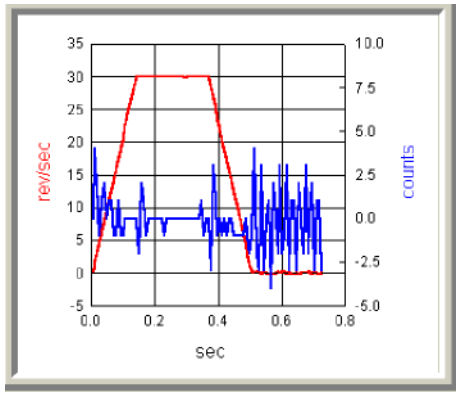
The final value to set is the “Integral Gain (KI)”. The KI value works to minimize steady state position error. This is most often needed to insure very exact positioning when at the new position after the move. It can also help minimize position error during the move.

Again start with a small value and work up.
The KI value affects how fast the position error is acted on; larger values provide faster response times. As the value gets larger you may notice an oscillation in the position




KI = 4000

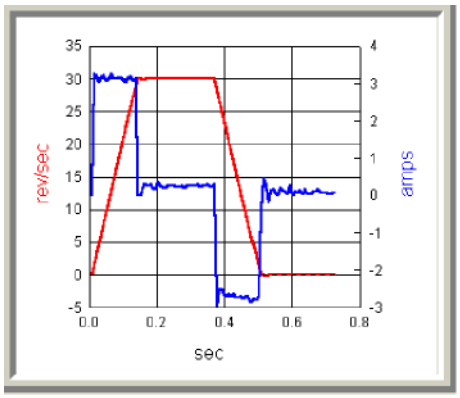



KI = 20000

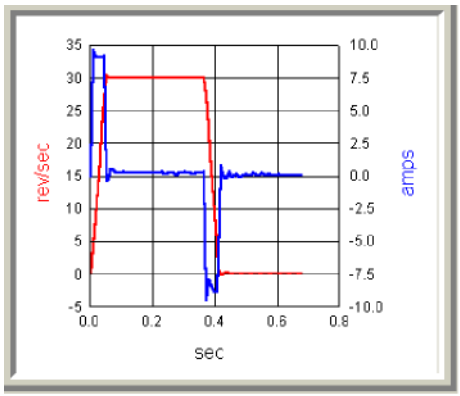
error. Adjust the KI value to give the best results without causing an oscillation in the system

### *8) Verify the Drive Current*

This can be done at any time during the tuning process to make sure the current supplied to the motor is not being limited by the drive. You might also want to see how much current is being required and make changes to the move profile.




Acceleration = 200 rev/sec$^2$




Acceleration = 600 rev/sec$^2$